Aspectos económicos y modelos de negocio del software libre

David Megías Jiménez (coordinador)
Amadeu Albós Raya
Lluís Bru Martínez
Irene Fernández Monsalve

PID_00145008

UOC
Universitat Oberta de Catalunya
www.uoc.edu

David Megías Jiménez

Ingeniero en Informática por la Universidad Autónoma de Barcelona (UAB). Magíster en Técnicas avanzadas de automatización de procesos por la UAB. Doctor en Informática por la UAB. Profesor de los Estudios de Informática y Multimedia de la UOC.

Amadeu Albós Raya

Ingeniero de Informática por la Universitat Oberta de Catalunya y por la Universidad de Andorra, especialidad en seguridad, redes y sistemas distribuidos. Actualmente, es profesor, coordinador y administrador de sistemas informáticos en un centro docente, miembro del Grup de Recerca en Seguretat Informàtica i Programari Lliure de la Universidad de Andorra, y consultor del máster universitario de Software libre que ofrece la Universitat Oberta de Catalunya.

Lluís Bru Martínez

Irene Fernández Monsalve

Licenciada en Ciencias Ambientales por la Universidad Autónoma de Madrid. Máster internacional en Software libre por la Universitat Oberta de Catalunya. Es socia fundadora de Haicku, S. Coop. Mad., empresa para la que trabaja en la actualidad en la gestión e implantación de software libre en Centros de Acceso Público a Internet en la Sierra Norte de Madrid.

Primera edición: septiembre 2009

Av. Tibidabo, 39-43, 08035 Barcelona
Diseño: Manel Andreu
Realización editorial: Eureca Media, SL
ISBN: 978-84-691-8694-7
Depósito legal: B-2.239-2009

Introducción

En nuestra sociedad, las industrias de las tecnologías de la información y del conocimiento (TIC), es decir, las telecomunicaciones, la electrónica (de producción y de consumo) y la informática han cobrado una importancia desconocida tiempo atrás. Se ha producido un ritmo extraordinario de innovación en estas industrias, acompañado de un proceso de convergencia entre ellas, manipulando, transmitiendo y reproduciendo la información con procedimientos comunes: las tecnologías digitales.

Pero, más allá de los cambios que se han producido en las industrias directamente relacionadas con las TIC, tienen mucha más importancia los cambios y mejoras que han influido en otras actividades económicas, especialmente las más tradicionales, donde la manera de proceder ha vivido una remodelación en profundidad. Las consecuencias en el tejido económico han propiciado la apertura de nuevas áreas de negocio que responden a nuevas demandas del mercado, inéditas tiempo atrás.

Ahora bien, ninguna ley económica ha cambiado y ninguno de los fenómenos económicos relacionados con las TIC son cualitativamente nuevos. Si acaso, lo que ha cambiado es la importancia relativa de determinados efectos económicos en nuestra sociedad, como por ejemplo las políticas de propiedad intelectual o de compatibilidad de los productos. Si nos centramos más concretamente en los efectos económicos relevantes en la industria del software, llegaremos a una conclusión parecida.

En este sentido, la industria del software ha constatado en los últimos tiempos la apertura y despliegue del software libre al público en general. El software libre se ha posicionado en distintos sectores del mercado de software genérico como una alternativa válida y viable, y de forma concluyente en algunos sectores especializados (por ejemplo, el software de servidor web). Esta situación ha incidido en la industria del software basada en el modelo propietario, y ha provocado la aparición de nuevos modelos de negocio relacionados con la difusión, el desarrollo, el soporte y la implantación del software libre.

La asignatura de *Aspectos económicos y modelos de negocio del software libre* tiene el objetivo de proporcionar los conocimientos necesarios para comprender y poner en práctica la economía del software libre mediante el estudio y el análisis de los aspectos económicos y de los modelos de negocio relacionados, así como de las oportunidades que ofrece este nuevo mercado.

En el primer módulo de la asignatura, se describa los conceptos económicos necesarios para estudiar la estructura económica de la industria del software en general, y en particular para articular un modelo de negocio a partir del software libre.

En el segundo módulo de esta asignatura, se describen de forma breve las características principales del mercado del software con relación a los modelos de negocio consolidados y cuáles son los clientes potenciales de los proyectos empresariales basados en software libre.

En el tercer módulo de la asignatura se presenta el software desde el punto de vista del negocio, centrando la atención en el mercado donde compiten las diferentes soluciones y en la estrategia empresarial para conseguir captar a los potenciales clientes.

En el cuarto módulo se estudian los modelos de negocio relacionados con el software libre. En un primer momento se analizan las clasificaciones propuestas por algunos autores de referencia. Después, se define una taxonomía propia en la asignatura, que considera las características económicas más relevantes del negocio

En el quinto módulo se analiza el desarrollo de software libre desde el punto de vista empresarial. Se consideran tanto las características más relevantes del proyecto de software como las particularidades relacionadas con la comunidad de usuarios de software libre y su gestión funcional y legal.

En el sexto módulo se presentan las principales ventajas e inconvenientes del modelo del software libre para la estrategia empresarial, considerando tanto la perspectiva del cliente como de la misma empresa y el modelo de negocio que explota.

En el séptimo y último módulo de la asignatura, se estudia el software libre como modelo económico, centrando la atención tanto en las bases fundamentales de su existencia como en las implicaciones actuales y futuras con relación al negocio basado en software libre.

Objetivos

Al cursar esta asignatura, el estudiante deberá lograr los siguientes objetivos:

1. Identificar y comprender los factores económicos más relevantes relacionados con el sector tecnológico y, especialmente, con el sector del software libre.

2. Entender los diferentes modelos de negocio explotables en la industria del software, así como sus principales características.

3. Profundizar en los diferentes modelos de negocio vinculados al software libre y en sus oportunidades de negocio.

4. Comprender y relacionar los factores económicos, técnicos y legales necesarios para crear y mantener una empresa basada en software libre.

5. Analizar casos de explotación de software libre en el sector privado y su relación con el mercado tecnológico.

6. Profundizar en las estrategias para desarrollar y explotar software libre con fines lucrativos.

7. Planificar y elaborar el diseño de negocios válidos y viables basados en la explotación del software libre.

Contenidos

1. Las bases del modelo
2. Las características del modelo del software libre
3. La validez y la viabilidad del modelo del software libre

Recursos

Asociaciones de empresas de software libre:

- Asociación Catalana de Empresas por el Software Libre: (www.catpl.org).
- Asociación de Empresas de Software Libre de Canarias (www.eslic.info).
- Asociación de Empresas de Software Libre de Euskadi (www.esle-elkartea.org).
- Asociación de Empresas Gallegas de Software Libre (www.agasol.org).
- Asociación Madrileña de Empresas de Software Libre (www.solimadrid.org).
- Open Source Business Organisations of Europe (www.obooe.eu).

Crear un negocio:

- Centro Europeo de Empresas e Innovación del Principado de Asturias: Guía para la creación de empresas y elaboración del Plan de Empresa (http://www.guia.ceei.es).

Conferencias (software libre y empresa):

- Open Source Business Conference (OSBC) (http://www.osbc.com/live/13/events/13SFO07A/SN441958/CC141932/ y http://www.osbc.com).
- WhyFLOSS (http://www.whyfloss.com/es/conference).

Datos de empresas:

- Cámara de Comercio (http://www.camerdata.es).
- Hoovers (http://www.hoovers.com).

Casos de estudio:

- Avanzada7: "Business models based on Asterisk: The case of Avanzada7" (http://www.whyfloss.com/es/conference/madrid08/getpdf/64).

- IBM migration to Free Desktops (http://www.channelregister.co.uk/2007/02/13/ibm_open_client).

- Liferay: "Liferay Enterprise Portal: The project, the product, the community and how to extend it" (http://www.whyfloss.com/es/conference/madrid08/getpdf/66).

- Openbravo: "Openbravo: keys to success in free software application development" (http://www.whyfloss.com/es/conference/madrid08/getpdf/49).

- Red Hat and JBoss: "Is Open Source viable in Industry? The case of Red Hat and JBoss" (http://www.whyfloss.com/es/conference/madrid08/getpdf/68).

- Varios casos (www.opensourceacademy.gov.uk/solutions/casestudies).

Blogs software libre, innovación y empresa

Galopini, R. *Commercial open source software* <http://robertogaloppini.net/> [Consulta: marzo 2009]

Oswalder, A. *Business model design and innovation blog.* <http://business-model-design.blogspot.com/> [Consulta: marzo 2009]

The 451 group. *451 caos theory: A blog for the enterprise open source community.* <http://blogs.the451group.com/opensource/> [Consulta: marzo 2009]

Bibliografía

Sieber, S. (2006). *Factores que influyen en la adopción del código abierto.* IESE <www.iese.edu/es/files/Art_Computing_Sieber_Factorescódigoabierto_May06_tcm5-5510.pdf> [Consulta: marzo 2009]

Sieber, S.; Valor, J. (2005). *Criterios de adopción de las tecnologías de información y comunicación.* IESE <www.iese.edu/en/files/6_15211.pdf> [Consulta: marzo 2009]

Dixon, J. (2007). "The Beekeeper: Crossing the Chasm Between the Cathedral and the Bazaar". *A Description of Professional Open Source Business Models.* <http://wiki.pentaho.com/pages/viewpageattachments.action?pageId=8346> [Consulta: marzo 2009]

Woods, D.; Guliani, G. (2005). *Open Source for the Enterprise: Managing Risks, Reaping Rewards O'Reilly Media.* <http://books.google.com/books?id=f4qJiKytvgC&dq=open+source+for+the+enterprise>

Nociones básicas de economía

Lluís Bru Martínez

PID_00145050

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

En este primer módulo, se presentan los principales conceptos relacionados con la economía de los productos y, especialmente, las particularidades del negocio de las tecnologías de la información y de la comunicación. Estos conceptos pretenden dar una base donde fundamentar las diferentes actuaciones y modelos de negocio que establece la política empresarial y que se verán más adelante.

En el primer apartado se presentan las nociones básicas del valor de un producto a partir de la demanda y de la oferta, y de la ventaja competitiva respecto de sus competidores como herramientas fundamentales para la viabilidad del negocio.

En el segundo apartado se presentan los principales efectos económicos que se relacionan con las características de los productos tecnológicos y del software en general. Se detalla cómo la empresa puede actuar sobre el mercado estableciendo una política de manipulación de estos efectos para crear un escenario que le sea el más favorable posible respecto de sus competidores.

Objetivos

Al finalizar este módulo, el estudiante debe alcanzar los objetivos siguientes:

1. Comprender los fundamentos básicos del funcionamiento entre demanda y oferta, especialmente los conceptos relacionados con la creación de valor.

2. Identificar y examinar las principales características económicas de la industria del software.

3. Profundizar y relacionar los efectos económicos ligados al mercado del software.

4. Identificar y analizar los efectos económicos susceptibles de transmitir valor o ventaja competitiva a los productos basados en software libre.

5. Profundizar en las políticas y estrategias de gestión del mercado de software libre.

1. La creación de valor

Para que un determinado negocio sea viable, es necesario que exista gente o empresas dispuestas, como clientes, a pagar por el producto o servicio propio que se les ofrece, y que estos pagos compensen a sus proveedores los gastos que conlleva atenderlos. Lo primero que vamos a plantear de una manera sencilla son los conceptos económicos fundamentales que se plantean en esta interacción entre la empresa que organiza un negocio y los posibles clientes de su producto o servicio.

1.1. La demanda de un producto

En primer lugar, debemos describir unas posibles reglas de comportamiento de las empresas y las familias a los que queremos convertir en clientes de nuestro negocio.

Un consumidor (si se trata de un bien de consumo) o una empresa (si se trata de adquirir maquinaria, materia prima, etc.) se plantearán comprar un determinado producto o servicio si la cantidad de dinero (el pago) que se les pide a cambio les parece razonable.

En esta situación, el posible comprador está realizando el siguiente razonamiento:

1) En primer lugar, le parece razonable pagar como mucho una cantidad de dinero V por obtener a cambio el producto o servicio que se le ofrece. Por lo tanto, si se le pide una cantidad de dinero P menor que V, considera que vale la pena obtener el producto. Así pues, para que alguien se plantee ser nuestro cliente, es necesario que se cumpla que

Valoración del producto – su precio = V-P > 0

Dicho de otra manera, una empresa no conseguirá que le paguen más de V por su producto o servicio. No obstante, esta circunstancia no garantiza que el cliente compre el producto.

2) En segundo lugar, el cliente comparará esta propuesta con las otras alternativas disponibles. Entre dos o más productos parecidos, el consumidor escogerá aquél en que la diferencia V–P sea mayor.

Ejemplo práctico

Una familia se plantea comprar un coche. La familia valora el modelo del fabricante A en 40.000 euros, Va= 40.000 euros, y el precio de venta es de 30.000 euros, Pa = 30.000. La familia valora menos el modelo del fabricante B; supongamos que en concreto lo valora por sus menores prestaciones (por ejemplo, es un vehículo de menor tamaño) en 35.000 euros, Vb= 35.000 euros.

La familia del ejemplo comprará el modelo del fabricante A aunque sea más caro, siempre que el coche del fabricante B se venda por más de 25.000; y viceversa, comprará el coche del fabricante B si es suficientemente barato, es decir, si su precio está por debajo de 25.000 euros:

Concluimos que:

Se compra el producto del fabricante A solamente si

$$Va–Pa = 40.000 – 30.000 > Vb–Pb = 35.000–Pb,$$

es decir, solamente si Pb > 25.000.

Se compra el producto del fabricante B solamente si

$$Va–Pa = 40.000 – 30.000 < Vb–Pb = 35.000–Pb,$$

es decir, solamente si Pb < 25.000.

La **demanda** de un determinado producto consiste en el conjunto de clientes que se consiguen para cada precio posible del producto en cuestión.

En nuestro ejemplo, si todas las familias valoran estos productos de la misma manera para precios por encima de 25.000 euros, no hay demanda para el producto del fabricante B, mientras que para precios inferiores, tenemos la demanda de todas las familias que valoran el producto de la misma familia que hemos considerado.

¿De qué depende el valor V que un posible cliente concede a un producto o servicio? En primer lugar, por supuesto, de la calidad intrínseca del producto para satisfacer las necesidades del cliente; pero también:

1) De la capacidad del cliente para valorar convenientemente el producto, lo que depende en gran medida de su formación y educación.

Será difícil que un cliente valore el sistema operativo GNU/Linux, por ejemplo, si ni siquiera sabe lo que es un sistema operativo, y ni se le ha ocurrido pensar que un ordenador no tiene por qué llevar instalado necesariamente el sistema operativo Microsoft Windows.

2) De la importancia de disponer de productos secundarios que complementen el producto principal que se nos está ofreciendo (un coche es más valioso si las carreteras son mejores, si es fácil encontrar gasolineras, y es menos valioso si las carreteras están congestionadas, si el transporte público es de calidad, si la gasolina se encarece, etc.)

3) De la difusión real del producto que se nos ofrece, es decir, de la cantidad de gente que también dispone de él: el teléfono o el correo electrónico es más valioso cuanta más gente lo utiliza.

1.2. La oferta de un producto

Por su parte, un empresario se dedicará a un determinado producto en la medida en que pueda obtener un beneficio razonable, que consiste en cubrir dos aspectos esenciales:

1) Los costes de atender a los clientes.

2) Lo que se ganaría dedicándose a otra actividad.

Ejemplo práctico

Digamos que una pareja decide abrir un bar. Al cabo de un año, han obtenido unos ingresos de 150.000 euros, y los costes de atender a la clientela, el alquiler del local, etc. han supuesto 120.000 euros. Vemos que el primer requisito se ha cumplido, dado que los ingresos han superado ampliamente los costes; un contable nos diría que tenemos beneficios, porque los ingresos cubren los costes.

Pero imaginemos que, para abrir el bar, esta pareja ha renunciado a sus respectivos trabajos como asalariados, que les reportaban unos ingresos anuales de 40.000 euros. Estos ingresos alternativos son lo que los economistas denominan el coste de oportunidad de montar el bar como negocio. Vemos que el segundo punto no se cumple en el ejemplo:

El negocio no es realmente provechoso porque

$$\text{Ingresos} - \text{Costes} = 150.000 - 120.000 = 30.000$$

$$< \text{Coste de oportunidad} = 40.000$$

Por supuesto, esta pareja puede seguir prefiriendo regentar su bar a trabajar como asalariados, y nos parecerá pues razonable su sacrificio en la cantidad de dinero de que disfrutan al final del año si les compensa la satisfacción de regentar su propio negocio. Nuestro punto es, en primer lugar, que no están llevando a cabo un buen negocio desde el punto de vista estrictamente monetario; y en segundo lugar, que nos parecerá razonable su decisión siempre y cuando sea una decisión lúcida, es decir, que aceptan conscientemente su merma de ingresos; no nos parecería razonable si no se hacen cargo de que van a disponer de menos dinero.

Para que abrir el bar sea realmente un buen negocio, se requiere que los beneficios que se obtengan excedan el "beneficio" de la actividad alternativa o coste de oportunidad.

Si los ingresos anuales del bar fuesen de 180.000 euros, por ejemplo, entonces sí que estaríamos hablando de un buen negocio:

Estamos ante un buen negocio porque

$$\text{Ingresos} - \text{Costes} = 180.000 - 120.000 = 60.000$$

$$> \text{Coste de oportunidad} = 40.000$$

Llegamos a la conclusión de que, para que un determinado negocio se mantenga, es necesario que los beneficios que se obtienen excedan los costes de oportunidad de dedicarse a otras actividades alternativas.

1.3. La creación de valor y la ventaja competitiva

A partir de lo que hemos visto hasta ahora, ya podemos plantear cuáles son los requisitos que se deben cumplir para que un negocio sea rentable.

En primer lugar es necesario que se cree valor, o sea, que la valoración V que los posibles clientes hacen del producto que se ofrece exceda sus costes:

Para que un negocio sea viable, es requisito imprescindible que la

Valoración del producto – Costes – Coste de oportunidad > 0

Solamente cuando esto se cumpla diremos que una empresa crea valor y puede ser viable, porque solamente en este caso podemos encontrar un precio que sea razonable tanto para el cliente como para la empresa.

Ejemplo práctico

Si el valor de un producto para un cliente es V=100 euros y atenderlo representa unos costes C = 60, podemos encontrar un precio satisfactorio para el cliente y la empresa, por ejemplo P = 80, de manera que el intercambio se produce de manera satisfactoria para ambos porque se cumple a la vez que

V – P > 0

y

P – Costes totales > 0

Que la condición V – C > 0 se cumpla, sin embargo, no garantiza que un negocio sea viable. Para ejemplificar esta situación se puede extender al ejemplo anterior del apartado 1.1 (relativo a la demanda de producto), pero desde el punto de vista de dos empresas rivales, que intentan atraer un cliente:

Ejemplo práctico

Tenemos dos fabricantes de automóviles que ofrecen dos modelos similares. Hemos visto que la familia valoraba uno de los coches en Va = 40.000 y el del otro fabricante en Vb = 35.000.

Imaginemos que los costes de fabricación de la empresa A son Ca = 20.000, mientras los de la empresa B son Cb = 10.000. Ambas empresas fabrican a unos costes muy inferiores a la valoración respectiva Va y Vb.

Por lo tanto, si no existiese un rival, indiscutiblemente cualquiera de las dos empresas sería viable como negocio.

Imaginemos que la empresa B decide vender sus vehículos al precio Pb = 18.000. La satisfacción del cliente es

$$Vb - Pb = 35.000 - 18.000 = 17.000.$$

La empresa A debe ofrecer una satisfacción mayor (o al menos similar) para captar clientela:

La empresa A capta a los clientes si:

$$Va - Pa > Vb - Pb = 17.000 \text{ solamente si } Pa < 23.000.$$

La empresa A tiene pues la posibilidad de captar clientes y cubrir sus costes. Pero fijémonos que esta empresa está a merced de su rival:

Así, si la empresa B decide rebajar sus precios por debajo de 15.000 (por ejemplo Pb=14.000), la empresa A no puede seguir captando clientes sin incurrir en pérdidas:

$$Vb - Pb = 35.000 - 14.000 = 21.000 \text{ y}$$

$$Va - Pa > Vb - Pb = 21.000 \text{ solamente si } Pa < 19.000, \text{ pero entonces}$$

$$Pa - Ca < 0 !$$

En este ejemplo, la empresa B dispone de una ventaja competitiva respecto a su rival, la empresa A. La consecuencia es que acaban sucediendo una de estas dos situaciones:

1) La empresa B acapara toda la clientela, como cuando fija Pb = 14.000, o bien

2) Las dos empresas se reparten la clientela, pero la empresa B gana más dinero con cada cliente:

Se reparten la clientela si Va – Pa = Vb – Pb, pero esto significa que Pa – Ca < Pb – Cb, por ejemplo si Pb = 18.000 y Pa = 23.000.

En definitiva, la empresa con una ventaja competitiva tiene asegurada su supervivencia y en cualquier caso ganar más dinero que sus rivales.

En el ejemplo anterior, la empresa B disfrutaba de una ventaja competitiva en costes: aunque su producto no era tal vez el más adecuado a las necesidades de los clientes, Va > Vb, era capaz de producir un producto razonable a unos costes muy inferiores a los de su rival.

Inditex

Un ejemplo interesante para el objetivo de esta asignatura es la empresa **Inditex** (propietaria de la cadena de tiendas de ropa Zara). La industria de la ropa de moda, en el que se desenvuelve esta empresa, es un sector altamente competitivo, donde las empresas pueden imitarse sin restricciones en sus diseños unas a las otras, y observamos pese a todo un altísimo nivel de inventiva, con nuevos modelos en cada temporada, año tras año (y por supuesto una gran cantidad de empresas que se dedican a esta actividad), y unos precios ajustadísimos. Por lo tanto, como clientes podemos disfrutar de las ventajas de una industria altamente competitiva e innovadora.

Pese a todo, Inditex consigue ampliar año tras año su cuota de mercado (es decir, acaparar una mayor parte de la clientela) gracias a su ventaja competitiva en costes, que según parece consiste básicamente en (1) detectar rápidamente los modelos que se venden mejor en una temporada determinada y (2) ajustar seguidamente la producción hacia estos modelos, de manera que sus costes son menores porque no produce ropa que no se venda, y vende mucha de la ropa de la que gusta ese año.

Y al parecer, llevar a cabo ambas cosas no es trivial, porque sus rivales no son capaces de imitar su comportamiento (al menos de una forma tan brillante).

Una empresa con una ventaja competitiva en costes consigue captar más clientes y obtener mayores beneficios porque puede vender sus productos más baratos.

Alternativamente, una empresa podría tener una ventaja competitiva en diferenciación, es decir, ofrecer un producto mejor valorado que el de sus rivales a un coste razonable.

La empresa Adobe

Posiblemente la empresa Adobe con su programa de software Acrobat, es un buen ejemplo de producto mejor valorado a un coste razonable.

Y esta mayor valoración superior puede ser general, en el sentido de que toda la posible clientela considera su producto de mayor calidad (sería el caso de los coches de marcas alemanas de prestigio, por ejemplo); o bien de nicho, es decir, que se trata de un producto especializado, adecuado para una clientela concreta (cualquier tienda de pueblo cumple este requisito: es una tienda orientada a una clientela concreta, a saber, los residentes en el pueblo, los únicos para los que resulta más cómodo comprar allí el pan o el periódico).

La ventaja competitiva en diferenciación permite a la empresa vender más caro sin perder por ello la clientela.

1.4. Recapitulación

Hemos visto de una manera simplificada en qué consiste, desde el punto de vista económico, crear un **negocio viable**. Consiste, en resumidas cuentas, en crear algún producto o servicio provechoso para la clientela, de manera que se pueda cobrar por ello manteniendo controlados los costes.

Desde el punto de vista de la posibilidad de crear un negocio a partir del software libre, la pregunta económica crucial es: ¿qué producto o servicio se puede cobrar a la clientela? Antes de entrar en ello, en el siguiente apartado veremos una serie de características económicas relevantes de la industria del software, necesarias para entender la respuesta a la pregunta anterior.

2. Características económicas de la industria del software

Como decíamos al principio, ninguna ley económica ha cambiado, y ninguno de los fenómenos económicos relacionados con las industrias de las tecnologías de la información y del conocimiento (TIC) son cualitativamente nuevos. Si acaso, lo que ha cambiado es la importancia relativa de determinados efectos económicos en nuestra sociedad. En particular, en las industrias relacionadas con las TIC, en la interacción de mercado entre las empresas y sus clientes, cobran una gran importancia una serie de fenómenos económicos que pueden distorsionar el funcionamiento de estos mercados. A continuación vamos a ver someramente los siguientes efectos:

1) Los costes de copia y distribución de la tecnología digital.

2) La economía de las ideas y la propiedad intelectual.

3) Complementariedades.

4) Efectos de red.

5) Productos compatibles y estándares.

6) Costes de cambio y clientes cautivos.

7) Políticas de compatibilidad y estandarización dentro de una plataforma y entre plataformas.

Un ejemplo reciente de este último apartado es la **compatibilidad entre plataformas y la política** que sigue al respecto la empresa de **software propietario Microsoft**, que ha provocado la intervención de la Comisión Europea en defensa de la **libre competencia** entre empresas. Por su importancia para el correcto desarrollo de los modelos de negocio basados en el software libre, analizaremos también brevemente el punto de vista de la Comisión Europea al respecto.

2.1. Los costes de producción, copia y distribución de la tecnología digital

La tecnología digital presenta una estructura de costes muy concreta: desarrollar un producto concreto es muy caro, requiere grandes inversiones, y no sirve desarrollarlo a medias.

Seguidamente, realizar copias de elevada calidad del producto desarrollado y distribuirlo resulta relativamente barato.

Por lo tanto, atender a un cliente adicional resulta muy barato; lo que resulta costoso es la inversión inicial que permitirá desarrollar un producto alrededor del cual organizar un negocio.

La aviación comercial

De manera similar, una empresa de aviación comercial debe hacer una gran inversión en un avión si quiere establecer una conexión frecuente entre dos aeropuertos; medio avión no sirve, se lo tiene que procurar entero. Pero seguidamente, atender a un cliente adicional, hasta que el avión esté completo, resulta muy barato para la compañía.

Por supuesto, la reducción enorme de los costes de copia y distribución de los productos y servicios desarrollados mediante tecnología digital ha provocado algunos cambios importantes en determinadas industrias.

La industria discográfica

Un ejemplo paradigmático era la industria discográfica que se basaba en el control de la copia (una copia de calidad similar se entiende; con las tecnologías analógicas, una copia en casete era de calidad de sonido muy inferior a un disco de vinilo o un CD) y distribución del producto (básicamente a través de tiendas especializadas).

2.2. La economía de las ideas y la propiedad intelectual

Las TIC se caracterizan por el hecho de permitir manipular, transmitir y reproducir información o ideas. Por lo tanto, el progreso de estas tecnologías tiene el efecto principal de **facilitar la difusión de las ideas y su uso**.

Las ideas presentan, como bien económico, la particularidad de ser **bienes no rivales:** el hecho de que una persona use una idea no quiere decir que las otras no puedan hacer uso de ella también.

Bienes no rivales

Si Pedro se come una manzana, Juan no la podrá comer. En cambio, en el caso de una receta de cocina, si Pedro la usa, Juan también la podrá usar.

Las industrias de las TIC dedican una gran cantidad de recursos económicos a **desarrollar nuevos conocimientos**, con la intención de sacar provecho económico de la explotación de estas ideas. Desde el punto de vista del interés del conjunto de la sociedad, cada vez que se produce un nuevo conocimiento, sea un descubrimiento científico, una nueva técnica, etc., la difusión de esta nueva idea plantea un dilema. Por una parte, es evidente que, una vez se dispone de este nuevo conocimiento, el interés de la sociedad es que se produzca la mayor difusión posible de esta idea. Ahora bien, las empresas que han desarrollado este conocimiento lo han hecho por tal de sacar un beneficio económico, y esto sólo lo pueden conseguir restringiendo el acceso a este nuevo

conocimiento. Sin una cierta **protección contra la difusión** inmediata de estos conocimientos, se corre el riesgo que las empresas no se atrevan a invertir dinero en búsqueda y desarrollo de nuevas ideas y conocimientos.

Las sociedades avanzadas han creado diferentes instituciones y mecanismos orientados a facilitar la creación de nuevos conocimientos científicos y técnicos. La creación científica se financia básicamente a través de recursos públicos. El desarrollo y financiación de conocimiento más práctico y aplicado, orientado a la creación de nuevas técnicas de producción y nuevos productos, en general se deja en manos del sector privado. En estos casos, lo que hacen las instituciones públicas es favorecer la actividad del sector privado mediante la protección de la propiedad intelectual, a través de la institución de una serie de figuras jurídicas, básicamente **los derechos de copia** (**copyright**), **las patentes** y **el secreto industrial**.

El *copyright* protege la expresión particular de una idea.

Supuestos de *copyright*

El ejemplo típico es el derecho que el autor de una canción o un libro tiene sobre su obra, de forma que nadie puede publicarla o distribuirla sin su consentimiento. La persona o empresa que hace un descubrimiento útil puede pedir que sea aplicada una patente, la cual prohíbe que, durante unos cuantos años (normalmente 20), se pueda hacer uso de este descubrimiento sin su consentimiento. Finalmente, tenemos la figura del secreto industrial: las empresas pueden mantener el secreto de este nuevo conocimiento, y reciben protección legal contra su robo. En este último caso, obviamente, el inventor no está protegido si otros llegan a hacer los mismos descubrimientos de manera independiente a través de su propio esfuerzo.

Si bien un uso adecuado de algunas de estas figuras de protección de la propiedad intelectual efectivamente pueden favorecer el progreso técnico y económico, desgraciadamente nos encontramos con dos problemas: que es muy discutible que todas estas figuras jurídicas realmente protejan el desarrollo de las ideas; y que en los últimos años muchas empresas hacen un uso espurio de las figuras jurídicas que podrían ser útiles. En lugar de proteger legítimamente la innovación que llevan a cabo, muchas empresas están usando los derechos de *copyright* y las patentes de que disponen como instrumentos anti-competitivos, para proteger su poder de mercado y hacer más difícil la entrada de rivales más innovadores.

En el caso del software, lo que ha permitido la aparición de sistemas propietarios es la facilidad de las empresas de mantener el secreto industrial por el hecho de que se puede distinguir entre el código fuente y el código binario del software. Podemos usar un programa, o sea, podemos conseguir que el hardware –sea un ordenador, un teléfono móvil, una consola de juego, un cajero automático, etc.– funcione con un programa informático, si incorporamos el código binario al ordenador sin disponer el código fuente. Por lo tanto, las empresas de software propietario tienen un modelo de negocio que consiste en cobrar dinero por proveer una copia del código binario de su software. La

consecuencia es que, sin conocer el código fuente, no podremos saber por qué el programa funciona de una manera determinada y no de otra, y desde luego no lo podremos modificar para que nos permita hacer otras cosas.

El secreto industrial (no revelar el código fuente), pues, permite a las empresas por una parte esconder a los rivales el producto desarrollado, y pese a todo, vender un producto a los consumidores (el código binario del programa informático).

El **software libre**, precisamente de manera completamente opuesta, está basado en **compartir el código fuente** del programa. Como veremos, esto implica desarrollar un modelo de negocio completamente diferente, basado en ofrecer un servicio: la capacidad de modificar y adaptar el software a las necesidades del cliente, a partir de la pericia y los conocimientos de que dispone el ingeniero informático.

El *copyright*, las patentes y la innovación

P. Tampoco es partidario de las patentes en el software...

R. Digamos que soy muy escéptico de que sirvan para lo que supuestamente están diseñadas. El software es una industria en la que la innovación es secuencial. Cada nuevo descubrimiento o mejora se construye sobre lo que se ha desarrollado antes, como si fuera una torre. Una patente aplicada en un determinado nivel de la torre frena avances posteriores. En la práctica funciona como un monopolio.

De una entrevista a Eric Maskin, premio Nobel de Economía 2007, publicada en *El País*, 29-06-2008.

Lectura recomendada

Podéis leer la entrevista completa en el artículo publicado en *El País*, 29-06-08 "Es difícil prevenir una burbuja"

¿Es verdad que un creador está realmente tan desprotegido sin la existencia de *copyright* o patentes sobre sus ideas? Muchos creadores parecen pensar así. El famoso cocinero Ferran Adrià, por ejemplo, en un diálogo con el gerente de la empresa Bimbo, publicada en *El País*, el 11 de agosto del 2006, decía:

"Una de las cosas que no está solucionada en este país es la protección de la creatividad. Te pueden copiar sin problemas. El I+D tiene poco sentido. En los restaurantes, igual."

...

"¡Que inventes algo y que al mes te lo copien! En la vida hay cosas que están mal hechas, que no funcionan, y ésta es una. Que tú estés trabajando durante años con mucha ilusión y que al mes venga alguien y lo presente sin romperse la cabeza..."

Lectura recomendada

Podéis leer el diálogo completo en el artículo publicado en *El País*, 11 de agosto de 2006.

¿Realmente es tan fácil copiar sus ideas? ¿Significa esto que su modelo de negocio no pueda funcionar? Que su negocio funciona es obvio. ¿Qué es lo que impide que Ferran Adrià se quede sin clientela?

1) En primer lugar, lo que realmente vende Ferran Adrià a sus clientes no es una idea (una receta) sino un plato cocinado. La idea, para ser consumida por sus clientes, debe estar incorporada a un plato cocinado concreto, igual que uno no compra en su día una idea de un coche, sino un coche concreto.

2) En segundo lugar, y ligado al hecho de que consumimos o usamos productos y servicios que concretan una idea, no basta con tener la idea a la vista, la "receta"; es necesario disponer, para poder convertirla en el plato cocinado, de la pericia y conocimientos y de las herramientas adecuados. Respecto a lo segundo (las herramientas), el mismo Adrià acostumbra a decir que los espectadores no deben pretender repetir los platos cocinados por él en su restaurante, porque en las cocinas caseras no disponemos de los utensilios adecuados. En casa nos recomienda cocinar cosas sencillas.

Por lo tanto, la inversión en las herramientas que permitirán reproducir la idea ya limita el número posible de imitadores, y por lo tanto el número de copias auténticas, es decir, platos efectivamente cocinados por profesionales, que pueden rivalizar con el suyo. Este es un punto fundamental a tener en cuenta en cualquier industria. Copiar la idea no es tan obvio, es decir, convertirlo en producto o servicio requiere unos conocimientos (sea la pericia que da la experiencia o el conocimiento adquirido por el estudio, o ambas cosas) y unas inversiones en maquinaria, herramientas, materias primas, etc. que limitan el nivel de rivalidad efectivo en la industria.

El técnico profesional

Esto es algo que se da posiblemente en cualquier actividad profesional. Podemos cambiar o ajustar tal vez los grifos de nuestra casa, pero posiblemente no dispondremos de las herramientas adecuadas de las que dispone un fontanero (es un gasto excesivo comprarlas para cambiar un grifo cada tantos años), si es que realmente nos creemos capacitados técnicamente para la labor.

3) En tercer lugar, como señala Maskin en el caso del software, y también es el caso del diseño textil y el desarrollo de software, la innovación culinaria se produce de manera secuencial y acumulativa: cada nueva receta no surge de la nada, sino que se basa en resultados previos. Pero esto es algo que el mismo Adrià ya lo señalaba en una serie de artículos suyos, escritos conjuntamente con Xavier Moret y publicados en agosto del 2002 en *El País*, reseñando sus viajes a distintos países:

"Actualmente, hemos adoptado los viajes como un método creativo; es decir, vamos a inspirarnos, a buscar chispas que nos den ideas, o ideas concretas de otras cocinas que puedan hacer evolucionar nuestra cocina.[...]Esta actitud, la de conocer lo que hacen los otros, creo que es vital en cualquier actividad en la que quieras evolucionar."

Así pues, la innovación no parece partir de cero, sino que, cada vez que propone una nueva receta, ésta se inspira en mayor o menor medida de la de sus predecesores, sea la cocina establecida en la tradición culinaria de su país, sea en cocinas de otros países. Su reputación de inventor de recetas y buen ejecutor de ellas (su reputación, construida sobre la base de la experiencia de los que han sido sus clientes en su restaurante) le permite disfrutar de lo que llamamos en el apartado 1.3. una ventaja competitiva en diferenciación, que le permite cobrar un mayor precio que otros cocineros (tal vez imitadores suyos) sin perder por ello su clientela.

Alternativamente, una empresa puede basar su ventaja competitiva en sus menores costes, como señalábamos anteriormente para el caso de Zara: tal vez no es la empresa más innovadora de su industria, pero se inspira o adapta los diseños de otras empresas con cierta gracia (es decir, a la gente le gusta vestir la ropa que se vende en sus tiendas) y es capaz de tener unos costes inferiores a los de sus rivales.

2.3. Complementariedades

Cuando hablamos de software, debemos tener presente que lo que valoramos en realidad no es el producto separadamente, sino que lo que valoramos es un conjunto de productos que se complementan entre sí; de hecho, el software no es nada más que una de las piezas del sistema que realmente nos sirve.

La existencia de complementariedades es frecuente en productos y servicios relacionados con las TIC.

Productos complementarios

Hay televisores de calidades muy diferentes, pero en realidad incluso el mejor de los televisores es un aparato perfectamente inútil si no disponemos de conexión con cadenas de televisión, de reproductor de DVD, etc.

La complementariedad de los equipos informáticos

Del mismo modo, no queremos simplemente disponer de un ordenador (entendiendo por el ordenador simplemente el objeto físico, igual como hablábamos del televisor) sino que también queremos disponer de los objetos físicos que complementan el ordenador, como por ejemplo impresoras, máquinas de fotos digitales, escáneres, etc. Y con todos estos objetos físicos tampoco nos basta, sino que además necesitamos el software. Disponer de todo lo que haga funcionar el ordenador (es decir, el sistema operativo), y seguidamente del software que denominamos aplicaciones, y que nos permiten emplear el ordenador para realizar diferentes tareas. Son ejemplos de software de aplicación un paquete de ofimática, el navegador de Internet, el correo electrónico, etc.

Por lo tanto, la complementariedad de los diferentes productos que conforma un sistema en cualquier tecnología digital (no sólo el ordenador) lleva a que cada elemento aislado del sistema no sirva realmente de gran cosa. Desde luego, esto quiere decir que pasa a ser fundamental que estas diferentes piezas se entiendan entre ellas y funcionen correctamente a la vez, es decir, la necesidad que los diferentes componentes sean compatibles entre ellos.

2.4. Efectos de red

Decimos que se dan efectos o externalidades de red cuando el valor de un producto o sistema para cada persona concreta que lo usa es mayor cuanta más gente lo usa. Las externalidades de red pueden ser de dos clases diferentes:

1) Directas.

2) Indirectas o virtuales.

La **externalidad directa** es tal vez más fácil de entender: muy a menudo, un producto nos resulta más valioso cuanta más difusión tiene, porque entonces podemos compartir la utilización con más gente.

Externalidad directa

Son ejemplos obvios de ello el teléfono, el fax, el e-mail, etc. Fijémonos que para que realmente podamos aprovechar esta masa de gente que también tiene un teléfono, es fundamental que el suyo y el nuestro sean compatibles (se entienden entre ellos). No nos servirá de gran cosa tener fax y que los otros también lo tengan si resulta que su fax no acepta o no entiende los mensajes que les envía el nuestro.

Como discutiremos con más detalle en el siguiente apartado, los posibles efectos de red se desaprovechan si no se produce un proceso de estandarización que asegura que los objetos en manos de personas distintas son compatibles entre sí, porque sólo entonces podremos realmente comunicarnos con mucha gente.

La telefonía móvil

En Estados Unidos, las distintas empresas que ofrecen servicios de telefonía móvil no se pusieron de acuerdo en utilizar el mismo sistema. Por lo tanto, en Estados Unidos un teléfono móvil es mucho menos útil que en Europa, donde la Comisión Europea promovió el uso de un mismo estándar en todos los países. La consecuencia inmediata es que la telefonía móvil está mucho menos extendida en Estados Unidos, en perjuicio de todo el sector, empresas y clientela.

Las **externalidades de red indirectas** son un efecto económico más sutil. Cuando un producto es de hecho un sistema hecho de diferentes piezas que se complementan, y cada una de ellas no es demasiado valiosa sin las otras, el valor de un producto acaba dependiendo de su popularidad, porque dispondremos de más complementos (o de piezas de más calidad) cuanta más gente esté interesada en el producto.

En cualquier caso, los efectos indirectos y los directos tienen algo en común: Nuevamente, es fundamental que las otras personas y empresas dispongan de **productos compatibles**.

En estos casos resulta fundamental, para que los mercados de estos productos despeguen, una de estas dos situaciones: o bien que la Administración pública intervenga; o bien que tome la iniciativa un agente económico con un poder suficiente como para modificar por sí solo las condiciones de mercado y como para disponer de suficientes recursos financieros para soportar los años de adaptación de los clientes.

Efectos indirectos y directos

Dos ejemplos de la importancia de estos efectos para el despegue de productos que presentan externalidades de red:

1) Los nuevos formatos de vídeo de alta definición. Las empresas fabricantes del nuevo diseño han buscado el compromiso de las grandes productoras de cine, que han asegura-

do que difundirán sus nuevas producciones en este formato. De esta manera, se garantiza a los clientes que dispondrán del complemento de los nuevos reproductores, el soporte diseñado para aprovechar la resolución superior de estos aparatos.

2) Podemos ver mediante el siguiente ejemplo que este efecto económico aparece en otros sectores, no sólo en las TIC. Fijémonos que no querremos comprar un coche que use los nuevos combustibles biodiesel (o sea, el elaborado a partir de aceites vegetales) si no podemos encontrar estaciones de servicio que lo provean; y a su vez, cada estación de servivio individual tiene poco interés en modificar sus surtidores y depósitos si no cree disponer de clientes, los fabricantes no se animarán a elaborar biodiesel, etc.

En estos casos, no nos hace ningún servicio directo (contrariamente a lo que pasa cuando la otra gente también tiene fax) que las otras personas también tengan coches que funcionan con biodiesel (o sea, no hay ningún efecto directo); ahora, sólo cuando exista una masa considerable de gente que tengan coches biodiesel, las estaciones de servicio verán el negocio de adaptar sus depósitos y surtidores al nuevo combustible. Podemos decir que, indirectamente, cualquier persona que compra un coche biodiesel hace un favor a los otros compradores de coches biodiesel.

Las externalidades indirectas explican pues la importancia para el crecimiento del uso de este nuevo combustible por el hecho de que la Administración pública subvencione los costes de fabricación; y también la importancia del acuerdo reciente entre **Acciona**, ahora mismo la empresa española más adelantada técnicamente en el proceso de fabricación de biodiesel, y **Repsol**, que posee la principal red de distribución de combustibles en España. El acuerdo entre estas dos empresas asegura que las estaciones de servicio dispondrán en un futuro cercano de carburante biodiesel. A su vez, ahora los fabricantes y concesionarios se animarán a vender coches biodiesel, porque podrán garantizar a los compradores que el abastecimiento de combustible no será complicado.

Cuando hablamos de productos y servicios que tienen las complementariedades y los efectos de red como características importantes, la consecuencia más importante que resulta de ello es que un producto no es viable si no se consigue una masa crítica suficiente de usuarios: por debajo de una cantidad determinada de usuarios, el producto no ofrece unas prestaciones suficientes que lo hagan valioso, y los posibles proveedores de productos complementarios no llevarán a cabo las inversiones necesarias para ponerlos al alcance de los clientes.

El formato VHS

La inercia hacia el uso de una versión puede eliminar pues la viabilidad de versiones alternativas, en principio técnicamente factibles. Los reproductores de vídeo **betamax** desaparecieron cuando todo el mundo decidió tener en su lugar reproductores **VHS**. Pese a que la cantidad total de familias que disponían de vídeos crecía año tras año, y que la cantidad de películas disponibles en vídeo también crecía, los propietarios de vídeos betamax no lo podían aprovechar porque la mayoría de nuevos títulos sólo salían en el formato VHS, mucho más popular; y los fabricantes dedicaron sus esfuerzos a mejorar los reproductores, a partir de un determinado momento, solamente las versiones VHS.

Otro peligro que se deriva de estos efectos es que una empresa establecida, que dispone de una base de clientes considerable, puede parar el funcionamiento normal de la competencia a través de acciones estratégicas que hacen difícil o imposible que los nuevos productos y servicios de los rivales consigan una masa crítica suficiente.

En el mundo del software, veremos acto seguido que la principal estrategia anti-competitiva consiste en hacer el producto de la empresa que domina el mercado incompatible con el producto de los rivales.

2.5. Productos compatibles y estándares

Podemos definir un estándar como el conjunto de especificaciones técnicas que permiten que las diferentes piezas de un sistema sean compatibles entre ellas.

Como hemos visto en los apartados anteriores, el valor de un producto depende en gran manera de la existencia de unos estándares aceptados:

1) Cuando un producto está hecho de diferentes elementos que se complementan.

2) Cuando los efectos de red son importantes.

En la industria de las TIC, con respecto al hardware (o sea, si hacemos referencia a los aparatos físicos) es evidente que el proceso de estandarización, afortunadamente, ha avanzado extraordinariamente. Hoy en día, casi todos los periféricos de un ordenador pueden conectarse a un puerto presente en el ordenador (por ejemplo, un puerto USB), y cuando compramos una impresora, pongamos por caso, sabemos que no debemos preocuparnos lo más mínimo: seguro que cuando lleguemos a casa podremos conectarla fácilmente a nuestro ordenador.

La obsolescencia de los componentes

Los que tengan una cierta edad, si hacen memoria, recordarán que las cosas no iban de este modo años atrás. Cualquiera de nosotros ha sufrido la experiencia de haber comprado una pieza o aparato electrónico o informático que ha quedado obsoleto simplemente porque ya no lo podemos conectar al resto de componentes de los que debería formar parte.

Y los más jóvenes se pueden hacer cargo de lo que decimos si piensan en todos los cargadores que nos vemos obligados a arrastrar a todas partes (el del móvil, el del ordenador portátil, etc.) porque el cargador de uno de estos aparatos no sirve para los otros (¡Muchas veces incluso cuando son productos fabricados por un mismo fabricante!). Si un día decidimos cambiar de móvil, sabemos que, desgraciadamente, ya podemos tirar el cargador, porque no nos volverá a ser de ningún uso.

2.6. Costes de cambio y clientes cautivos

Muy a menudo, hay productos pensados para ofrecer un servicio similar que desgraciadamente no son compatibles entre ellos. Fue el caso en su día de los discos de vinilo y los discos compactos, y lo fue después de los reproductores de vídeo de formato VHS y DVD.

Objetivamente, podemos afirmar en estos dos ejemplos que una de las tecnologías es claramente superior a la otra y, por lo tanto, si debiéramos escoger entre las dos tecnologías partiendo de cero, no tendríamos ninguna duda sobre cuál emplear.

Debido a la existencia de complementariedades, sin embargo, para todos aquellos que usaban la tecnología anticuada, el cambio les resultó en su momento muy costoso. Quienes tenían discos de vinilo y querían pasar a tener compactos, antes que nada debían comprar un reproductor de compactos, y se veían forzados también a comprar de nuevo los discos, si querían reproducirlos con la nueva tecnología.

En general, debido a las complementariedades y a los efectos de red en el mundo de las TIC, cambiar de una versión del producto a otra incompatible es un proceso costoso, hasta el punto de que, posiblemente, seguiremos usando la tecnología antigua durante mucho tiempo salvo que la mejora de calidad nos parezca muy importante.

Desde luego, en el mundo de la informática y del software en particular estos costes de cambio pueden ser importantes. Incluyen los costes de aprender nuevos programas cuando estamos acostumbrados a una versión determinada. De ahí la tendencia de los programadores a hacer que los nuevos programas tengan un aire y un funcionamiento parecido a los programas que ya conocemos.

Programas similares

El procesador de textos OpenOffice imita a Microsoft Word, que a su vez imitaba un programa anterior, Wordperfect, que a su vez hacía lo propio con Wordstar (es decir, en cada caso, con el procesador de textos más popular en su momento); Microsoft Excel imita Lotus 1-2-3, que a su vez imitaba otro programa anterior, Visicalc. Y así podríamos encontrar muchos otros ejemplos.

Dados estos costes de cambio de pasar de un producto a otro si se dan incompatibilidades, las empresas establecidas, con una base de clientes sólida, tienen la tentación de aumentar artificialmente estos costes de cambio, de hacer más difícil a los clientes cambiar de producto o de proveedor.

Los costes de cambio

Tiempo atrás, los antiguos monopolios de telefonía, al surgir empresas rivales, intentaron forzar a sus clientes a cambiar de número de teléfono si querían cambiar de proveedor (la idea era que los clientes no querrían incurrir en el coste que implica comunicar a todos los conocidos el cambio de teléfono).

Del mismo modo, en el mundo del software, las empresas establecidas tienen la tentación de hacer poco compatibles sus productos con el de los rivales.

2.7. Políticas de compatibilidad y estandarización dentro de una plataforma y entre plataformas

Como hemos visto, que sean compatibles las diferentes partes que forman un producto, y que diferentes productos sean compatibles entre ellos, es fundamental para que sean mucho más funcionales. Por esto, es importante el establecimiento de estándares que permitan hacer **productos compatibles** entre ellos.

Muy a menudo, el **proceso de estandarización** se produce a partir del hecho de que el formato de una parte imprescindible de un sistema es adoptado por todo el mundo. Esta parte imprescindible que marca el proceso de estandarización es a veces denominada una plataforma.

A veces estos procesos de estandarización son el resultado del trabajo de organismos creados con el objetivo de definir estos estándares. Pueden ser organismos estatales o supraestatales, o creados por los miembros de la industria.

En el mundo del software, existen diferentes estándares establecidos con alguno de estos procedimientos, como por ejemplo todos los protocolos de comunicación que rigen la transferencia de información en Internet.

Otras veces, en cambio, una empresa de la industria controla una parte de la industria.

Sony y Phillips

Estas dos empresas consiguieron imponer por medio de la fuerza de los hechos su tecnología de creación de discos compactos. Y esto es así en la medida en que hoy en día todas las casas discográficas distribuyen su música con este formato digital, todos los aparatos musicales están pensados para reproducirlo, etc.

En el mundo del software, obviamente, el ejemplo principal de plataforma en el sentido que hemos planteado es el sistema operativo Microsoft Windows, presente en la gran mayoría de ordenadores, ya sean personales o servidores.

Es importante entender los intereses que mueven al propietario de un producto que se ha convertido, de una manera u otra, en una plataforma. En particular, veremos qué intereses guían las políticas de compatibilidad de su producto con los productos que lo complementan (políticas de compatibilidad dentro de una plataforma) y con los productos que son potenciales rivales del suyo (políticas de compatibilidad entre plataformas).

2.7.1. Políticas de compatibilidad y estandarización dentro de una plataforma

Dentro de una plataforma, un abanico de aplicaciones más grande puede hacer más valiosa la plataforma en dos sentidos: los clientes sacan más provecho de la plataforma (y por esto estarán dispuestos a pagar más); y los creadores de aplicaciones, a su vez, verán más posibilidades de negocio (al haber una base de clientes potenciales más grande) y por esto harán aplicaciones que funcionen en esta plataforma; esto, a su vez, atraerá a más clientes, etc.; de manera que se crea un círculo virtuoso que fomenta la difusión de este producto.

Así pues, más aplicaciones complementan la plataforma y la hacen más valiosa. En principio, el patrocinador de la plataforma debería tener interés en abrirla a los creadores de aplicaciones (y de hecho, Microsoft suele sostener que sigue una política abierta, porque enseña las partes del código de software de Windows (los API) que los creadores de aplicaciones deben conocer para que sus productos funcionen con Windows.

El promotor, sin embargo, tendrá intereses contrapuestos:

1) Si también tiene aplicaciones que le dan mucho rendimiento, le interesa empeorar el rendimiento de los productos rivales, haciéndolos en el peor de los casos incluso incompatibles con su plataforma.

2) También puede tener miedo de que determinadas aplicaciones, a su vez, puedan convertirse en nuevas plataformas en torno a las cuales desarrollar las otras aplicaciones, sin depender de la plataforma que controla.

Microsoft y Java

Esto es lo que pasó en su día con el navegador Netscape y con el lenguaje de programación Java: Microsoft llevó a cabo maniobras anti-competitivas contra este software porque le preocupaba que, a partir de su desarrollo, pudieran reemplazar a Windows como la plataforma de software de los PC.

Esto último nos anticipa, en cierta medida, el comportamiento que podemos prever que tendrá el propietario de una plataforma establecida como estándar de hecho, cuando tiene ante sí otros productos que le pueden ganar la posición de privilegio de que dispone, y que pasamos a discutir.

2.7.2. Políticas de compatibilidad y estandarización entre plataformas

Hemos visto más arriba que, debido a los costes de cambio, la cuota de clientes de que dispone la empresa que controla la plataforma puede ser una barrera a la entrada para los rivales, cuando hay efectos de red, si la empresa hace incompatible su producto con los de los rivales. Naturalmente, quienes salen perdiendo con estas tácticas anticompetitivas no son sólo las empresas rivales, sino también la sociedad en su conjunto, que ve reducidas de manera inmediata las posibilidades entre las cuales escoger, y a la larga, la calidad de los productos disponibles, porque hay menos empresas que se atrevan a dedicar recursos a la innovación y mejora de los productos.

Productos incompatibles, tácticas incompetitivas

El ejemplo más conocido de esta clase de comportamiento es el de Microsoft con respecto a sus dos productos estrella, **el sistema operativo Microsoft Windows** y **el paquete de ofimática Microsoft Office.** Claramente, Microsoft hace todo lo que puede por evitar que sea compatible con otras plataformas (en particular con el sistema operativo GNU/Linux, por ejemplo). En esta misma línea, Microsoft ha seguido sistemáticamente una política de incumplimiento de los diferentes estándares creados por la industria informática, creando su propia versión del estándar, sin documentar correctamente los cambios que introduce. Muy a menudo, cuando hay programas y aplicaciones que, aparentemente, no funcionan correctamente, es porque la plataforma no cumple los estándares aprobados por la industria.

El conflicto que las diferentes autoridades que defienden los intereses de la sociedad (tanto de Estados Unidos como de la Unión Europea) libran frente a Microsoft tiene básicamente que ver con esta actividad intencionada de manipulación del proceso de estandarización de una tecnología, modificando la capacidad de comunicación y de interoperar entre diferentes plataformas de información.

2.7.3. Políticas públicas respecto al software

Seguidamente, discutimos brevemente algunas políticas públicas que pueden favorecer el funcionamiento correcto de los mercados de software, y en particular, que pueden permitir que el software libre pueda competir en igualdad

de condiciones como una alternativa válida y viable al software propietario, en aquellos casos en los que este último software parte con la ventaja de disponer de una masa de usuarios ya establecida.

Defensa de la competencia

En primer lugar, las administraciones públicas deben garantizar el correcto funcionamiento de la **competencia** en el mercado del software.

La principal acción de las autoridades de la competencia debe ser garantizar que no se creen incompatibilidades artificiales (o sea, que no tengan una explicación técnica) entre diferentes plataformas tecnológicas.

El conflicto que la Comisión Europea tiene actualmente con Microsoft es debido a que esta empresa manipula el grado de compatibilidad entre diferentes productos, modificando la capacidad de comunicación y de interoperar entre diferentes plataformas de software, en este caso, la comunicación entre los sistemas operativos que gestionan servidores informáticos y los que gestionan ordenadores personales.

La Comisión Europea pide a Microsoft que facilite a todo el mundo (en particular a los fabricantes de servidores informáticos y a los programadores) los protocolos de información del sistema operativo Windows, para que los otros sistemas operativos sean compatibles con este sistema, o sea, para que todos los otros sistemas operativos puedan comunicarse e interoperar con los servidores que funcionen con el mencionado sistema operativo.

Naturalmente, la intención de Microsoft es aprovechar que ya dispone de una fuerte implantación del sistema operativo Windows, subiendo artificialmente los costes de cambio a otro software por parte de sus clientes.

Políticas de adopción y apoyo del software libre. Forzar el cumplimiento de los estándares

Hemos visto la importancia de los efectos de red en las TIC y la necesidad de una masa crítica de usuarios para que un software sea viable. Debido a estos efectos de red, las grandes empresas pueden ejercer un liderazgo en la implantación del software libre. Si la Administración pública y las grandes empresas (por interés propio o como servicio a la sociedad) fomentaran el software libre en sus organizaciones, podrían crear la suficiente masa crítica como para que los ciudadanos vieran facilitado el uso del software libre.

Mucho del software propietario que se usa hoy en día en estas organizaciones podría ser reemplazado fácilmente por software libre con prestaciones parecidas o mejores, y que el único obstáculo que hay es el coste de cambio por cada usuario particular a causa de la carencia de una masa crítica suficiente.

Es importante el efecto de red a que daría lugar esta política de las organizaciones mencionadas, en particular por los efectos de red indirectos que se generarían: la adquisición de software libre por parte de estas grandes organizaciones crearía una fuente de negocio importante para las empresas informáticas que orientaran su modelo de negocio en torno al software libre y a la provisión de los servicios de pericia informática que son complementarios a su implantación.

En cualquier caso, en primer lugar, estos organismos deberán seguir un proceso de adquisición de software que exigiera el cumplimiento de unos determinados protocolos y estándares de compatibilidades. Establecer unos procedimientos de adquisición de software y de servicios informáticos adecuados por parte de las administraciones públicas, por ejemplo, requeriría previsiblemente la creación de una agencia pública encargada de asesorar a los diferentes departamentos gubernamentales. Estos organismos pueden implementar diferentes mecanismos que fomenten el uso del software libre dentro de los organismos públicos.

Resumen

El negocio de las tecnologías de la información y de la comunicación tienen particularidades concretas que influyen en el modelo económico del negocio y, consecuentemente, en el mercado.

Más allá de la creación de valor en los productos y de la gestión para conseguir una ventaja competitiva sobre sus competidores en el mercado, la empresa puede incidir en los efectos económicos del mercado mediante el establecimiento de una política estratégica concreta:

- Mientras que los costes de producción son elevados, los costes de copia son mínimos.
- La explotación de ideas y la salvaguarda de la propiedad intelectual.
- La explotación de complementariedades del producto.
- El efecto red del producto, ya sea relacionando la valía con la utilización masiva o como promotor indirecto de complementos.
- La compatibilidad entre productos rivales.
- El control de los costes de cambio ante la evolución del producto y la cautividad de los clientes.
- El establecimiento de políticas sobre la compatibilidad y estandarización dentro y entre plataformas.

Con todo ello, las particularidades del software libre le permiten establecer un nuevo formato de negocio que rompe las políticas habituales de un mercado tecnológico muy tradicional en cuanto al posicionamiento de la competencia.

Bibliografía

Boldrin, Michele; Levine, David (2008). *Against Intellectual Monopoly*. Cambridge: Cambridge University Press. <http://levine.sscnet.ucla.edu/general/intellectual/againstfinal.htm>

Jaffe, Adam B.; Lerner, Josh (2004). *Innovation and Its Discontents*. New Jersey: Princeton University Press

Lerner, Josh; Tirole, Jean (2002). "Some Simple Economics of Open Source".*The Journal of Industrial Economics* (pág. 197-234).

Perens, Bruce (2005, octubre). "The emerging economic paradigm of open source". *First Monday.* Special Issue #2: Open Source. <http://firstmonday.org/issues/special10_10/perens/index.html>

Shapiro, Carl; Varian, Hal (1999). *Information Rules: A Strategic Guide to the Network Economy*. Boston: Harvard Business Press

Prensa

Adrià, Ferran (2002, 1 de agosto). "Cazadores de ideas". *El País*.

"Aquí unos amigos" (entrevista a Ferran Adrià, 2008, 19 de julio). *El País*.

"Does IT matter?" (2004, 1 de abril). *The Economist*.

"Es difícil prevenir una burbuja" (entrevista a Eric Maskin, 2008, 29 de junio). *El País*.

<http://people.ischool.berkeley.edu/~hal/people/hal/NYTimes/2004-10-21.html>

"El tomo ha muerto, viva la red" (2007, 22 de julio). *El País*. Negocios.

"Prince vuelve a enfurecer a la industria musical" (2007, 15 de julio). *El País*.

"Star Turns, Close Enough to Touch"(2007, 12 de julio). *New York Times*.

Varian, Hal (2004, 21 de octubre). "Patent Protection Gone Awry". *New York Times*.

El mercado del software

Lluís Bru Martínez

PID_00145048

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

En este módulo se presentan las principales características del mercado del software en general, y de cómo encaja el modelo del software libre en este mercado.

En el primer apartado veremos que es bastante habitual en nuestro entorno poder acceder a productos que sean de libre distribución o directamente gratuitos, y conoceremos cuál es el funcionamiento particular de este negocio.

En el segundo apartado se presenta brevemente el mercado objetivo del software, así como los medios más habituales a partir de los cuales los potenciales clientes consiguen el producto.

Objetivos

Al finalizar este módulo, el estudiante debe alcanzar los objetivos siguientes:

1. Comprender las características del mercado de productos de libre acceso.

2. Entender la relación entre el software libre y la explotación de modelos de negocio paralelos.

3. Entender las implicaciones del abastecimiento del software en el concepto de negocio.

4. Profundizar en los paradigmas de desarrollo de software, así como relacionarlos con las características del software libre.

1. Negocios con características similares al software libre

Ahora que ya hemos visto las principales nociones de economía, podemos replantearnos la pregunta que dejamos pendiente en el apartado 1.4 "Recapitulación" del módulo 1:

Si el software libre es libre, o sea, por definición todo el mundo puede tener acceso a este software –eventualmente con coste cero–, ¿cómo va a ser posible que haya informáticos (y empresas informáticas) que se dediquen a programar software libre? ¿Podemos confiar en que en el futuro se dedicarán recursos (dinero y el tiempo de personas) a su mantenimiento y desarrollo?

1.1. ¿Es realmente tan chocante que el software pueda ser libre?

O dicho de otra manera, ¿es realmente tan infrecuente que un producto sea de libre distribución o eventualmente gratis? Tenemos a la vista, si prestamos un poco de atención, algunos modelos de negocio basados en ofrecer gratuitamente un producto a su clientela.

En líneas generales, cualquier empresa que tiene como negocio ejercer de intermediaria entre otras empresas y sus clientes debe decidir qué política de precios seguir. Y posiblemente la mejor opción es renunciar directamente a ganar dinero con alguno de estos clientes.

Productos gratuitos

En España tenemos televisiones como Antena 3, Cuatro, Telecinco y La sexta que ofrecen gratuitamente la señal televisiva al espectador. Por supuesto, el negocio de estas televisiones consiste en vender publicidad, es decir, en ejercer de intermediarios entre las empresas que quieren dar a conocer su producto y la posible clientela (por ejemplo, los telespectadores que verán la publicidad colocada antes, durante y después de la retransmisión de Cuatro de un partido de fútbol).

Diferentes modelos de negocio basados en oferta gratuita

Si una televisión quiere conseguir ingresos publicitarios, debe asegurar a sus clientes de pago (las empresas que colocan anuncios publicitarios en las emisiones) el mayor número posible de telespectadores; y la mejor manera de conseguirlos consiste en permitir gratuitamente la recepción de la señal televisiva.

De la misma manera, si **Adobe** quiere conseguir clientes para su producto que sirve para elaborar ficheros .pdf, Adobe Acrobat Professional, tiene sentido ofrecer gratis la versión simplificada de este software, Adobe Acrobat Reader; de esta manera, Adobe puede asegurar a sus clientes de pago que realmente los demás usuarios podrán leer los documentos que elabore.

De igual forma, **Amazon**, además de ser una librería que vende a través de Internet, ha convertido su página web en una plataforma desde la cual pone en contacto a sus clientes con librerías de segunda mano, que ofrecen libros usados con un descuento. Al consultar la disponibilidad de un título, vemos la oferta de Amazon propiamente y la de las otras librerías. En este caso, Amazon ofrece gratuitamente a los clientes la posibilidad de consultar la base de libros disponible, mientras cobra a las librerías por su servicio de intermediación. A las razones comentadas anteriormente para seguir esta política de precios, se añade que es conveniente que Amazon gane dinero con las ventas de las otras librerías, porque en caso contrario podría tener la tentación de ofrecerles un servicio distorsionado (para garantizar la venta de sus propios libros frente a los libros de sus rivales expuestos en su página web).

Alternativamente, una empresa puede ofrecer gratuitamente un producto al cliente, pero ligado a otro producto, que es el que se pretende vender. Así, tenemos el siguiente ejemplo:

> "Todo aquel que haya comprado esta mañana el semanal británico *Mail on Sunday* se ha llevado una copia gratuita del nuevo trabajo de Prince, *Planet Earth* (Planeta Tierra). En total han sido vendidos 2,9 millones de ejemplares."
>
> [...]
>
> "*Planet Earth* también será distribuido gratuitamente a los que acudan a algunos de los 21 conciertos que el músico de Minneapolis ofrecerá en Londres desde el 1 de agosto hasta el 21 de septiembre en el Arena 02."
>
> *El País*, 15 de julio de 2007.

Lectura recomendada

Podéis leer el artículo completo publicado en *El País*, 15 de julio del 2007 "Prince vuelve a enfurecer a la industria musical".

Como vemos, en un caso es muy posiblemente el periódico quien compra el derecho a regalar copias junto a su publicación (una forma de publicitar el periódico), mientras que en segundo caso, el artista renuncia a ganar dinero con la distribución de las copias del disco (en contra de los esfuerzos de discográficas y tiendas de discos, que quieren mantener su modelo de negocio a toda costa), para concentrarse en conseguir ingresos mediante sus conciertos. (Otra noticia, esta vez del *New York Times*, señala que este mismo músico realiza conciertos exclusivos en locales de pequeño aforo, en el que la entrada se vende, con comida, por 3.000 dólares (12 de julio del 2007, "Star Turns, Close Enough to Touch").

Lectura recomendada

Podéis leer el artículo completo publicado en *El País*, 12 de julio del 2007 "Star Turns, Close Enough to Touch".

1.2. El software como parte de un producto

El software es solamente un componente más de un producto (por muy importante que sea esta pieza en particular), una pieza o complemento del producto total que queremos obtener, y de lo que queremos disponer es de todas las piezas conjuntamente, por ejemplo, del ordenador y del software a la vez.

Debido a esto, grandes empresas multinacionales del sector informático como IBM y Sun Microsystems financian informáticos que trabajan en el desarrollo de software libre. Su razón egoísta (en el sentido de pensar básicamente en un incremento de sus beneficios) es que piensan que de esta manera aumentarán las ventas de los productos y servicios complementarios por los que cobran a sus clientes.

Del mismo modo, las principales empresas fabricantes de teléfonos móviles (Nokia, Motorola, Siemens, Samsung, etc.) se asociaron (y dedican recursos económicos) para crear el consorcio Symbian, que se encarga de crear software libre pensado como programa que permite el funcionamiento de los teléfonos móviles que fabrican. De este modo, todos los fabricantes de teléfono móvil usan una misma plataforma (el mismo sistema operativo), basado en **el sistema operativo GNU/Linux**, que es lo suficientemente flexible para que cada uno de los fabricantes seguidamente diseñe un modelo de teléfono móvil diferente del de los rivales, incorporando mejoras y variaciones que les permitan captar clientes (teléfonos que son también máquina de fotos, que permiten

enviar correos electrónicos, etc.). Cada empresa modifica el aspecto que tiene la pantalla del teléfono para adaptarlo mejor a las prestaciones que ofrece, porque puede acceder al código fuente del programa que hace funcionar el aparato telefónico. De esta manera, se fomenta la innovación y mejora del producto, porque las empresas confían en conseguir nuevos clientes creando una máquina (el teléfono) que funcione mejor que la de los rivales.

El hecho de que grandes empresas multinacionales hayan incorporado completamente el software libre como herramienta de sus actividades garantiza, pues, el futuro desarrollo de este software. Incluso garantiza que, por iniciativa propia, haya ingenieros informáticos que se dediquen a desarrollar software libre. Como señalan Lerner y Tirole (2002), estos ingenieros pueden mostrar su pericia profesional a las empresas de este sector participando en la mejora de este software, lo cual hará que sean muy buscados por las empresas del sector informático, y por lo tanto les permitirá mejorar sus perspectivas laborales.

1.3. Abastecimiento de software. Distribución

Que el software sea libre no quiere decir que no puedan surgir empresas dedicadas exclusivamente a la actividad de abastecimiento de productos y servicios informáticos relacionados.

Para empezar, un posible negocio es distribuir software libre. Además de vender discos compactos que contienen el software libre, también ofrecen apoyo técnico a los consumidores y empresas que decidan usar software libre (Red Hat es el ejemplo más conocido de empresa que ha desarrollado esta línea de negocio). Por lo tanto, la empresa ofrece su experiencia y conocimiento del software al cliente, asegurándole el apoyo técnico que pueda necesitar.

Si lo pensamos bien, este modelo de negocio acaso no sea tan infrecuente. Por ejemplo, podemos interpretar que la editorial Aranzadi ha organizado un modelo de negocio muy similar.

Aranzadi

Aranzadi ofrece a sus clientes (profesionales del derecho) una fuente exhaustiva de información jurídica. Además, les ofrece el apoyo técnico necesario por poder procesar de manera eficaz toda esta información.

La información ha estado disponible libremente siempre (las leyes se publican en el Boletín Oficial del Estado, y cualquier bufete de abogados está suscrito a él). Pero organizar de manera útil la información es complicado; este es el servicio que estas editoriales proporcionan a sus clientes. Y por supuesto, estas empresas han incorporado las tecnologías digitales al servicio de sus clientes, como podemos ver en la siguiente noticia de prensa:

Los despachos de abogados y de expertos fiscales siguen decorados con metros y metros de solemnes tomos jurídicos. Pero estos son cada vez más un mero adorno. La mayor parte de los profesionales del Derecho se han decantado ya por internet para acceder a la documentación necesaria para su trabajo, una auténtica revolución propiciada por grandes editoriales jurídicas como Corporación El Derecho y que supone un ejemplo de éxito en las nuevas tecnologías.

Corporación El Derecho provee de información jurídica a los fiscales (a partir de un concurso del Ministerio de Justicia) y de información fiscal de base a la Agencia Tributaria.

El País, 22 de julio del 2007.

Lectura recomendada

Podéis leer el artículo completo publicado en *El País*, 22 de julio del 2007 "El tomo ha muerto, viva la red".

1.4. Abastecimiento de software. Servicio

En términos más generales, el ingeniero informático que trabaja con software libre es un profesional similar al cocinero, al mecánico de coches, al fontanero, o al abogado.

Un bufete de abogados trabaja con unos conocimientos e ideas de legislación que son tan libres, tan a disposición de todo el mundo, como lo pueda ser el software libre. Claramente, su modelo de negocio consiste en conseguir ingresos a partir de un producto complementario, que es su pericia, o sea, su conocimiento profundo de la legislación, su capacidad de organizar la información depositada en las leyes de manera adecuada para defender los intereses de su cliente, cosas que su cliente no tiene porqué ser capaz de hacer.

En definitiva, el abogado incorpora las ideas adecuadas al producto adecuado para su cliente (la defensa de sus intereses).

De manera parecida, el ingeniero informático que trabaja con software libre ofrecerá a sus clientes su pericia, su capacidad de satisfacer las necesidades de los clientes de organizar de una manera determinada la información, de procesar los datos, a partir de las posibilidades intrínsecas del software libre disponible, o, si es el caso, desarrollando código adicional.

De manera que vemos cómo un determinado sector económico (los servicios de abogacía) presenta incluso diferentes estratos de información (corporación, el derecho y Aranzadi en un nivel, los bufetes de abogados en otro), que da lugar a múltiples modelos de negocio que conviven a la vez.

2. ¿Quiénes necesitan software?

2.1. El software, una necesidad básica en cualquier empresa

¿Quiénes son los clientes de las empresas de software? Hoy en día, potencialmente cualquier empresa. Como señala Nicolas Carr en "IT doesn't matter", las TIC se han incorporado como una herramienta imprescindible para todas las empresas, igual como hoy en día todas las empresas están conectadas a la red eléctrica para iluminar las oficinas y proveer energía a sus máquinas; todas disponen de teléfono; o todas utilizan coches y camiones por las carreteras para trasladar sus materias primas y sus productos.

Lectura complementaria

N. Carr (2004, 1 de abril). "Does IT matter?". *The Economist*. <http://www.nicholasgcarr.com/articles/matter.html>

Cuando Carr decía en su artículo que "las TIC ya no cuentan", lo que quiere decir es que una empresa ya no tendrá una ventaja competitiva por el hecho de utilizarlas, puesto que todas las empresas tendrán acceso a ella.

Reservas de billetes en línea

Un caso muy citado al respecto son las compañías de aviación comercial que desarrollaron los primeros software de reservas de billetes; en su momento, disponer de este software les proporcionó una ventaja enorme respecto a sus rivales. Hoy en día, todas las empresas de aviación comercial disponen de una página web desde la que realizar la reserva y compra de un billete de avión; por lo tanto, este software ya no proporcionará una ventaja a una empresa sobre las demás.

Esta evolución en el uso de las TIC puede suponer una ventaja para el desarrollo del software libre, en la medida en que reduce la posibilidad de que las empresas se dejen llevar por la ilusión de que disponer de software propietario para sus procesos internos le puede proporcionar una ventaja competitiva. En la medida en que cualquier empresa puede disponer de software con capacidades similares, posiblemente lo mejor es utilizar un software libre, que pueda aprovechar los desarrollos llevados a cabo en otras actividades y adaptarlos a las necesidades concretas de la empresa.

2.2. Paradigmas de desarrollo del software

Si hemos dicho en el apartado anterior que todas las empresas hoy en día necesitarán usar las TIC, y en particular necesitarán usar software, ¿de qué manera puede obtener una empresa el software que va a necesitar en sus procesos productivos?

Siguiendo la clasificación desarrollada por Bruce Perens en "The emerging economic paradigm of open source", podemos clasificarlas de la siguiente manera:

Lectura obligatoria

B. Perens (2005). *The emerging economic paradigm of Open Source*. <http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1470/1385>

1) El modelo de Microsoft o Adobe (el modelo "Retail" según Perens), en el que una empresa se dedica por su cuenta a desarrollar software y lo vende empaquetado a sus clientes.

Por lo tanto, visto desde el punto de vista del cliente, se despreocupa del desarrollo del software y se limita a comprarlo ya acabado.

Las consecuencias del modelo al detalle (*retail*)

Naturalmente, este desarrollo de software suele tomar la forma de software propietario (donde el proveedor no enseña el código a sus clientes). Desde el punto de vista de quien adquiere este software, el primer problema evidente es que no está pensado para sus necesidades concretas (pues evidentemente tiene que venderse muy homogéneamente para que pueda interesar a clientes distintos). Otro problema posiblemente grave es, como vimos anteriormente, el peligro de quedar atrapado por el proveedor, que dificultará el cambio a otro software, recuperar determinadas bases de datos, etc. A la inversa, pero con consecuencias parecidas, se corre el peligro de que la empresa proveedora desaparezca y deje por lo tanto de proveer los servicios requeridos de mantenimiento y mejora del software.

2) El modelo donde la empresa necesitada de software lo desarrolla por su cuenta, bien con informáticos pertenecientes a su plantilla, bien contratando a una empresa informática especializada para que se lo desarrolle (el modelo "In-House and Contract" según Perens).

El coste del desarrollo

Por supuesto, esta forma de desarrollo del software necesario para una empresa puede resultar carísimo, y puede llevar a repetir partes de la programación que ya han desarrollado y que se hubiesen podido aprovechar.

En los dos últimos modelos de desarrollo en la clasificación de Perens, las empresas buscan otras empresas con las que colaborar en el desarrollo del software que necesitan.

3) En este modelo el consorcio elabora un software que no es libre (es decir, que no estará a disposición de las empresas que no participen en su desarrollo).

4) En el último modelo, las empresas del consorcio desarrollan software libre, es decir, con el código abierto a cualquier otra empresa, aunque no participe en su desarrollo.

Las ventajas claras del último es la posibilidad de aprovechar las mejoras de la comunidad de programadores que se cree alrededor del proyecto, con la consiguiente reducción de costes de desarrollo.

Por supuesto, el desarrollo de software libre no les resultará gratuito a las empresas del consorcio, que deberán financiar un grupo inicial de programadores. El peligro de un consorcio (tanto de software propietario como libre) es que se produzca una falta de liderazgo en el desarrollo del proyecto, porque ninguna empresa quiera comprometerse a asegurar su desarrollo, que impida su ejecución (bien desde el principio, bien cuando los sucesivos desarrollos impliquen nuevos costes).

Resumen

En nuestro entorno más inmediato se configura un escenario donde confluyen múltiples modelos de negocio con diferentes políticas para alcanzar sus objetivos. Desde la promoción directa del producto en sí, hasta el abastecimiento de productos gratuitos para que el cliente acceda a un nuevo mundo de productos y servicios complementarios.

El negocio del software libre adopta esta última forma particular de mercado, pudiendo establecer negocios paralelos y complementarios a partir de su promoción. Muchas son las empresas y multinacionales que hoy día han adoptado un posicionamiento claro para favorecer el desarrollo del software libre, especialmente considerando que el software es un producto básico para cualquier empresa y que el modelo de desarrollo del software libre les ofrece garantías para alcanzar estos objetivos.

Bibliografía

Karminski, D. (1999). "Core Competencies: Why Open Source Is The Optimum Economic Paradigm for Software". <http://www.doxpara.com/read.php/core.html> [Consulta: febrero 2009]

Perens, B. (2005). "The Emerging Economic Paradigm of Open Source". Publicado en *First Monday , Special Issue #2, 3/10/2005*. Cambridge: Cambridge University Press. <http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1470/1385> [Consulta: febrero 2009]

Prensa

"Prince vuelve a enfurecer a la industria musical" (2007, 15 de julio). *El País*.

"Star Turns, Close Enough to Touch" (2007, 12 de julio). *New York Times*.

Software como negocio

Irene Fernández Monsalve

PID_00145051

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

En este módulo vamos a prestar atención a la visión más "clásica" del software como negocio. Nos centraremos en el punto de vista del software propietario, dejando para un módulo posterior el estudio de las posibilidades adicionales que presenta el software libre en este escenario. Aunque algunos de los aspectos que trataremos se vuelven irrelevantes al aplicar estrategias de software libre, otros seguirán vigentes en gran medida.

Repasaremos algunos de los factores clave a la hora de diseñar un negocio en torno al software, como **la elección de actividad principal** y el enfoque general de la empresa (vender productos o servicios), aspectos de **comercialización** y **marketing** (cómo elegir un mercado, y cómo dirigirse a él), y la **definición de sus productos o servicios** (qué tipo de productos o servicios se van a desarrollar, y cómo se posicionarán).

Objetivos

Al finalizar este módulo, se deben alcanzar los objetivos siguientes:

1. Adquirir una visión global de las posibilidades de negocio en torno al software.

2. Conocer los modelos tradicionales de empresas de software.

3. Entender las características económicas y las diferencias entre las empresas de productos y las de servicios.

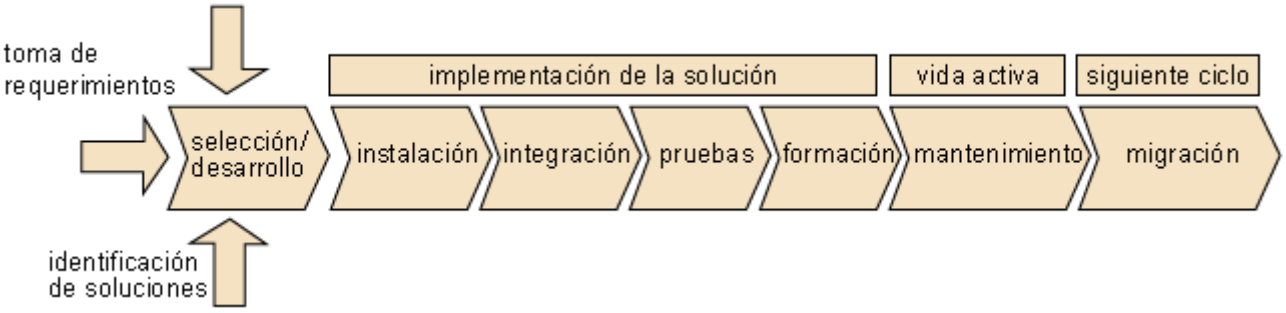
4. Identificar los factores clave que debe tener en cuenta una empresa de software para posicionar sus productos en el mercado.

1. Posibilidades de negocio en torno al software

Tanto las personas individuales como los entornos corporativos cuentan con unas necesidades de software que generan múltiples oportunidades de negocio.

La tarea central para satisfacer estas necesidades será la creación de ese software, el trabajo puro de desarrollo. Sin embargo, ahí no acaban las necesidades a explotar, sino que no hacen más que empezar. Una vez se dispone de un producto, surgen una serie de necesidades relacionadas, de consultoría, instalación, configuración, mantenimiento, soporte y formación, por las que ciertos clientes (principalmente otras empresas) estarán dispuestos a pagar.

A lo largo de todo el proceso de adopción de una tecnología, desde la identificación de necesidades, la decisión de construir o comprar, hasta el fin de su vida útil, se generan múltiples necesidades a las que distintas empresas pueden dar respuesta:



Proceso de adopción de una tecnología *(elaborado a partir de Carlo Daffara. "Sustainability of FLOSS-Based economic models". II Open Source World Conference.* Málaga. Disponible en: http://www.cospa-project.org/Assets/resources/daffara-OSWC2.pdf)

Por otra parte, el propio proceso de creación del software se puede entender de dos maneras: como la creación de un producto, o como la prestación de un servicio. La elección entre ambas será determinante a la hora de definir el funcionamiento de la empresa, y el potencial de generación de ingresos que presente, dando lugar a modelos de negocio muy distintos.

Esta disyuntiva –desarrollar software como producto o como servicio– refleja a su vez la primera cuestión que una empresa consumidora de software tendrá que evaluar a la hora de adoptar una solución tecnológica: comprar un producto estándar, empaquetado, u obtener un desarrollo a medida.

Podemos distinguir, por lo tanto, las siguientes actividades empresariales centradas en el software:

- Desarrollo de aplicaciones
  - Como producto: soluciones estándar (*shrink-wrapped*)

  - Como servicio: desarrollos a medida

- Prestación de servicios en torno a 1 o más aplicaciones
  - Asesoría
  - Selección
  - Instalación
  - Integración
  - Formación
  - Mantenimiento y soporte
  - etc.

- Software como servicio (*software as a service*)

Esta clasificación no pretende ser exhaustiva ni excluyente, es decir, muchas empresas implementarán modelos híbridos que les permitan ofrecer soluciones integrales a sus clientes.

Las características de las empresas de software, así como sus dinámicas de negocio, variarán mucho dependiendo de las actividades en las que se centren, como veremos a continuación, pero cualquiera de los modelos tienen potencial para generar tanto empresas viables como altos beneficios.

1.1. Empresas prestadoras de servicios

Tal como se ha comentado anteriormente, las empresas podrán implementar diferentes actividades al mismo tiempo, especializándose en unos o más aspectos de la cadena del proceso de adopción de una tecnología.

En este sentido, para las empresas que incluyen diversos servicios entre su oferta comercial podemos distinguir dos tipos de especializaciones: vertical y horizontal.

1.1.1. Especialización vertical

En general, las empresas que tengan como principal actividad el desarrollo tenderán a presentar una **especialización vertical**. Si su estrategia de negocio se centra en los desarrollos a medida, de forma natural sus actividades incluirán el resto de servicios relacionados, como **instalación**, **integración**, y **formación**. Pero, como veremos más adelante, aquellas empresas que hayan escogido una estrategia de software como producto harán bien en explotar también el resto de servicios asociados, como manera de asegurarse un flujo constante de ingresos.

	Paquete 1	**Paquete 2**	Paquete 3	Etc.
Desarrollo	**X**	**X**		
Instalación	**X**	**X**		
Integración	**X**	**X**		
Certificación	**X**	**X**		
Formación	**X**	**X**		
Mantenimiento y soporte	**X**	**X**		
Migración	**X**	**X**		

Especialización vertical (basada en Daffara. "Sustainability of FLOSS-Based economic models". *II Open Source World Conference*. Málaga. Disponible en: http://www.cospa-project.org/Assets/resources/daffara-OSWC2.pdf)

Resulta interesante destacar que una empresa que invierte una cierta cantidad en licencias de software espera invertir otro tanto en concepto de servicios relacionados como mantenimiento y soporte, así como en actualizaciones. De esta manera, la venta de productos a un cliente empresarial abrirá las puertas a obtener contratos de servicios con ese mismo cliente, y por lo tanto, un flujo de ingresos más constante a lo largo del tiempo.

1.1.2. Especialización horizontal

Por otro lado, las empresas que explotan las necesidades derivadas del uso general de productos de software a menudo abarcarán servicios en torno a varios paquetes, centrándose en una o varias de las fases de la adopción de una tecnología.

	Paquete 1	**Paquete 2**	**Paquete 3**	**Etc.**
Selección / Desarrollos a medida				
Instalación				
Integración				
Certificación	**X**	**X**	**X**	**X**
Formación	**X**	**X**	**X**	**X**
Mantenimiento y soporte				
Migración				

Especialización horizontal (basada en Daffara. "Sustainability of FLOSS-Based economic models". *II Open Source World Conference*. Málaga. Disponible en: http://www.cospa-project.org/Assets/resources/daffara-OSWC2.pdf).

Aunque existen empresas especializadas en la formación o en el soporte, a menudo las empresas de servicios abarcarán varias de las fases descritas, generando tipologías como, por ejemplo, la de la consultoría (centrándose en la selec-

ción, la asesoría, y/o certificación), o los proveedores de soluciones completas, que abarcan todas las categorías, incluyendo desarrollos a medida, e incluso la provisión de hardware.

Las empresas creadoras de distribuciones GNU/Linux siguen un modelo de **prestación de servicios con especialización horizontal**.

Ejemplo de especialización horizontal

Canonical, creadora de la distribución basada en **Debian Ubuntu**, realiza un trabajo de selección e integración horizontal, abarcando un sistema operativo completo junto con diversas aplicaciones, con el objetivo principal de proporcionar una distribución fácil de usar, instalar y configurar bajo el lema, "linux para seres humanos". Sin embargo, y dado el carácter libre de Ubuntu, los ingresos de Canonical provienen de servicios relacionados, como soporte, formación y certificación.

Estas empresas orientadas a servicios a menudo evidenciarán que sus clientes prefieren recibir soluciones completas y tratar con un único proveedor de soluciones tecnológicas. Para poder ofrecer este tipo de servicios integrales, a menudo es necesario contar con una potente infraestructura y capacidad técnica, lo que limitará la entrada de empresas medianas y pequeñas, que por sí mismas no serán capaces de dar respuesta a todas las necesidades.

Una solución común es que la empresa de servicios subcontrate las partes que no puede gestionar por sí misma. Otra solución que resulta bastante interesante es el "modelo de consultoría piramidal" que propone Daffara (*sustainability of FLOSS-Based economic models*) y que explicamos a continuación.

Web recomendada

Para más información acerca del "modelo de consultoría piramidal", podéis consultar: http://www.cospa-project.org/Assets/resources/daffara-OSWC2.pdf)

Como norma general, se puede decir que en el campo del soporte y el mantenimiento informáticos se cumple la regla 80/20: el 80% de las consultas son fáciles, y pueden solucionarse de forma inmediata. El 20% restante, en cambio, presenta problemas importantes, y requerirá el 80% del esfuerzo. Por lo tanto, una pequeña o mediana empresa de servicios podría abarcar un alto número de clientes, haciéndose cargo del 80% de sus incidencias, y cobrando por el servicio una cantidad moderada. Para solucionar el 20% restante, requerirá los servicios técnicos de las empresas creadoras del software, a las que previsiblemente tendrá que pagar más de lo que cobra por cada uno de sus clientes, pero menos de lo que ingresa por el conjunto de ellos.

Este modelo generará una cooperación sustentable entre las empresas desarrolladoras, con especialización vertical y las empresas proveedoras de soluciones integrales. Las primeras podrán alcanzar a más usuarios gracias a las consultoras horizontales, que supondrán además una fuente importante de ingresos. Las segundas lograrán manejar una amplía base de clientes y ofrecerles soporte de alta calidad sobre una gama de productos, manteniendo un negocio rentable siempre que su base de clientes sea suficientemente alta.

1.2. Empresas desarrolladoras: ¿crear productos o servicios?

Como hemos dicho anteriormente, una empresa que quiera centrarse en la actividad de desarrollo tendrá dos grandes líneas entre las que elegir: podrá generar **productos estándar**, empaquetados para vender a un mercado de masas (los llamados ***shrink-wrapped***), o podrá generar **desarrollos a medida**, específicos a cada cliente según sus necesidades.

La primera opción tiene el potencial de generar grandes márgenes de beneficios, aunque estos serán difíciles de mantener en el tiempo, y presenta unas barreras de entrada que pueden resultar infranqueables. La segunda supone una opción mucho más intensiva en cuanto a mano de obra, y con márgenes de beneficios mucho menores, aunque presenta más posibilidades de generar fuentes de ingresos constantes en el tiempo, y ser menos sensible a cambios en el entorno macroeconómico.

Vamos a analizar en detalle los aspectos diferenciadores de estas dos opciones.

Economías de escala y posibilidad de grandes márgenes de beneficios

Lectura obligatoria

M. Cusumano (2004). *The Business of Software* (cap. 1, "The Business of Software, a Personal View").

El proceso económico de creación de software tiene unas particularidades que no vemos en otros sectores, que le llevan a tener unos **retornos positivos de escala** enormes.

Por un lado, las compañías comerciales deben invertir grandes cantidades de dinero en desarrollo antes de tener una versión comercial de un producto para lanzar, y a menudo deben reinvertir otro tanto cada dos o tres años para mantener su flujo de ingresos constante. Al intentar generar un producto estándar, este desarrollo implica un alto riesgo, ya que no hay seguridad alguna de poder recuperar la inversión a través de ventas posteriores. Sin embargo, cuando se dispone de un producto finalizado, el coste marginal de cada copia adicional vendida es cercana a cero. La primera copia del software creada es muy cara, pero el resto no cuesta prácticamente nada.

Esto proporciona unas economías de escala, en el lado de la oferta, enormes, que se unen a unas economías de escala también importantes en el lado de la demanda: tanto por el tiempo invertido en adquirir destrezas en el uso de una aplicación, como por la posible incompatibilidad de formatos, cambiar de un producto a otro supone una tarea difícil y costosa. Como consecuencia, cuanto más grande sea la base de usuarios de un producto, más fácil es que esta base crezca y perdure en el tiempo. De esta manera, en el mercado del software se llega a situaciones de "el ganador se lleva todo" (*the winner takes it all*), pudiendo generar enormes beneficios, e impidiendo al mismo tiempo la entrada de nuevas empresas a estos mercados.

Ejemplos de empresas generadoras de productos estándar

Entre las empresas que han aprovechado estas fuertes economías de escala, encontramos algunas de las principales empresas del sector de software, como **Microsoft** copando los mercados de sistemas operativos de escritorio, y **Oracle** con su adquisición de **PeopleSoft** en el 2005. Sin embargo, también existen pequeñas empresas, Independent Software Vendors (ISV), que explotando nichos específicos han conseguido establecer negocios viables. Como ejemplos podemos citar el "Pretty Good Solitaire" desarrollado por la microempresa de dos trabajadores **Goodsol Development Inc.** (uno de los juegos de solitario con más éxito), o "HomeSite" un editor de HTML desarrollado por la microempresa Bradbury Software en 1995, que fue comprado por Allaire Corp. (Más tarde Allaire fue adquirida por Macromedia, que en el 2005 fue a su vez absorbida por Adobe.)

En contraste, una empresa centrada en el desarrollo a medida no disfrutará de las economías de escala del software estándar. Cada nuevo cliente requerirá un desarrollo específico, y por lo tanto una inversión en tiempo y esfuerzo costosa, a pesar de que este tipo de empresas tenderán a reutilizar sus desarrollos siempre que les sea posible.

1.2.1. Necesidad de inversión inicial

A la hora de crear una empresa centrada en la idea tradicional de producto surge un problema importante: la **necesidad de una inversión inicial**. Durante las primeras fases de la empresa, dedicadas al desarrollo, no habrá un flujo de ingresos, pero sí de gastos, hasta que las primeras versiones del software estén listas para ser comercializadas. Además de los gastos directamente derivados del desarrollo, habrá que tener en cuenta los gastos necesarios ligados al marketing y a la comercialización. Ante este problema, hay dos tipos de soluciones: **atraer inversión externa**, **o empezar con otro tipo de actividad empresarial** que genere ingresos suficientes para permitir el desarrollo simultáneo del producto.

Las **empresas de desarrollos a medida** implicarán **mucho menos riesgo** y podrán comenzar su actividad con una inversión mucho más pequeña (el desarrollo sólo se inicia después de tener un contrato firmado), pudiendo evitar la búsqueda de inversores externos.

En la literatura financiera se suele discutir más sobre aquellas empresas que financian sus desarrollos a partir de inversiones de capital riesgo, ya que resultan más atrayentes. Este tipo de financiación permitirá un crecimiento más rápido, siendo éste uno de los factores importantes para el éxito según Cusumano. (Michael Cusumano, *The Business of Software*)

Reflexión

En este punto, resulta interesante hacer la siguiente reflexión: ¿Con qué parámetros contamos a la hora de juzgar el éxito de una iniciativa empresarial? Los inversores, así como las publicaciones financieras, considerarán exitosas aquellas empresas que consigan presentar beneficios año tras año, y probablemente, aquellas que demuestren crecimiento. Una compañía cuyo tamaño se mantenga en el tiempo, y su cuenta de resultados no muestre beneficios no atraerá la atención de la literatura financiera ni de inversores. Sin embargo, una empresa de este tipo puede haber sido muy exitosa a la hora de crear puestos de trabajo de calidad, y a la hora de mantenerlos en el tiempo. Para muchas personas emprendedoras, éste puede suponer su objetivo principal.

Conseguir suficiente inversión externa puede suponer un escollo infranqueable, pero aun cuando se hace posible, presentará ciertas desventajas que será necesario valorar. La presencia de inversores supondrá una presión sobre las decisiones de gestión de la empresa, y requerirá que la empresa genere beneficios suficientes para poder devolver la inversión y proporcionar ganancias. Esta situación limitará la autonomía y capacidad de decisión de las personas fundadoras.

La otra opción tampoco resultaría fácil. La empresa tendría que orientar su actividad a los servicios, intentando que éstos generasen suficientes ingresos como para permitir el desarrollo simultáneo del producto. Como veremos más adelante, conseguir esto con la prestación de servicios será difícil, ya que el margen de beneficios será menor, y tanto la ausencia de economías de escala como la presencia de competencia limitan la capacidad de mantener precios suficientemente altos.

En este sentido, el software libre irrumpe con nuevas características que modifican este escenario. La posibilidad de **recortar los costes** gracias a la colaboración de personas voluntarias, así como los nuevos esquemas de difusión y comercialización que brinda esta colaboración suponen una importante disrupción de estos escenarios, teniendo el potencial de disminuir de forma considerable la inversión inicial necesaria.

Veremos más sobre estos aspectos en los módulos siguientes.

1.2.2. Mantenimiento del flujo de ingresos

Sin duda, una pregunta fundamental para cualquier empresa será no sólo cómo conseguir ingresos de forma puntual, sino como mantenerlos en el tiempo. Mientras que para las empresas centradas en la prestación servicios, la continuidad será la norma (en general, si los clientes están satisfechos, continuarán necesitando los servicios de forma recurrente), en aquellas empresas centradas en la producción de soluciones estándar, el mantenimiento de un flujo constante de ingresos encontrará diferentes problemas.

1) Ciclos del software

Cusumano, en *The Business of Software*, compara el proceso de escribir un producto de software exitoso con el de escribir un *best seller*. Conseguirlo genera enormes beneficios, pero además de resultar muy difícil, sólo los genera de forma puntual. El ciclo de vida natural de un producto de software comercial lo llevará finalmente a perder su capacidad de generar ingresos.

En un principio, las primeras versiones tendrán varios fallos, y su funcionalidad no se ajustará finamente a las necesidades de sus usuarios. Esto permitirá a la compañía creadora mantener sus ingresos en el tiempo mediante el lanzamiento de nuevas versiones que incorporen progresivamente mejoras en el producto, tanto por la solución de errores como por disponer de mucha más información de requerimientos obtenida como *feed-back* de sus usuarios y clientes.

Previsiblemente, si las nuevas versiones del producto presentan suficientes mejoras y resultan más atractivas que las anteriores, continuarán generando ingresos. Sin embargo, una vez los usuarios perciben que tienen una aplicación

suficientemente buena, desaparece la motivación por pagar una nueva versión. De forma análoga, intentar mantener los ingresos de un *best seller* mediante secuelas sólo tiene una efectividad limitada.

Existen estrategias para combatir estas tendencias y mantener un flujo de ingresos a través de licencias de versiones sucesivas, las más comunes a costa del consumidor. La incompatibilidad total o parcial entre versiones sucesivas del propio producto, unidas a intensas campañas de difusión de éste, llevarán a una nueva situación de economías de escala en el lado de la demanda a favor de la versión más reciente, que obligará a muchos usuarios a cambiar aunque el producto anterior satisfaga sus necesidades.

Sin embargo, por la naturaleza de algunos productos de software, la actualización constante se hace necesaria, por las necesidades cambiantes de los propios usuarios.

Un ejemplo ilustrativo: las aplicaciones de contabilidad, laborales y de manejo de impuestos.

Dado que la legislación sobre impuestos y materia laboral cambia a menudo, los usuarios necesitarán una actualización de su aplicación cada vez que esto ocurra, por lo que los ingresos se pueden mantener constantes en el tiempo gracias a los ajustes coyunturales en el sistema financiero y al marco legislativo.

Por otra parte, una vez la idea inicial ha sido explotada y analizada, se abrirá el camino para que otras empresas empiecen a producir un software análogo, sin necesidad de dedicar tiempo a I+D ni a la toma de requerimientos. Si consiguen hacer el producto más rápidamente, quizás simplificándolo y manteniendo sólo las funcionalidades justas, serán capaces de competir por el mismo mercado a un mejor precio. Una vez entren suficientes empresas en ese mercado, generando productos intercambiables entre sí (*commoditization*), se llega a una situación particular: en ausencia de otros factores de diferenciación, los consumidores comprarán el producto más barato, generando una situación de alta competencia.

Este fenómeno es común a cualquier tipo de producto, y en el software debería ser posible también. Sin embargo, ciertos factores protegen las empresas dominantes de este proceso, que en una situación ideal contribuiría a una mayor difusión tecnológica, y reportaría beneficios a los usuarios (aunque haga más difícil para las empresas el logro de grandes márgenes de beneficios). Como hemos apuntado antes, existen unas fuertes economías de escala en el lado de la demanda, por lo que no será tan fácil que los usuarios perciban los productos competidores como realmente sustituibles. Por otro lado, el uso de formatos propietarios genera una importante situación de cautividad de la que es complicado salir.

En este sentido, el software libre aparece como una fuerza impulsora de una situación de **bienes libres perfectamente intercambiables**: la aparición de un producto similar que se distribuye de forma libre, o eventualmente gratuita

hará más difícil mantener altos los ingresos derivados de licencias, y puede ser una de las pocas formas de lograr romper las inercias de cautividad que genera el software propietario.

Software libre como tecnología disruptiva

El término *tecnología disruptiva*, acuñado en 1999 por Clayton M. Christensen, se refiere a innovaciones que por su bajo precio y prestaciones, o por enfocarse a un nuevo tipo de clientes, consigue desplazar soluciones anteriores del mercado. El software libre podría suponer, de esta manera, una tecnología disruptiva, dada la eventual posibilidad de obtenerlo de forma gratuita, y dada su capacidad de contribuir a la generalización del uso del software a través de las brechas tecnológicas actuales.

La transformación del sector del software en un escenario de bienes intercambiables (*commoditization*), aunque limitaría en gran medida las posibilidades de mantener altos beneficios mediante licencias, podría abrir nuevos mercados, generando un ecosistema de necesidades en torno al nuevo producto intercambiable y de amplia adopción.

2) Dependencia de ciclos económicos

Las compañías tradicionales de software, centradas en productos, pueden generar enormes beneficios, pero también pueden sufrir enormes caídas y pérdidas en ciclos económicos desfavorables. A pesar de contar con negocios y productos asentados, del 2000 al 2002 muchas compañías de software perdieron del 80 al 90% de su valor, incluso Microsoft sufrió una pérdida de dos tercios de su valor (Michael Cusumano, *The Business of Software*).

Durante periodos económicos desfavorables, el consumo caerá, y los productos de software serán de los primeros en notar estos efectos. Los usuarios simplemente dejarán de comprar software, pudiendo afectar en gran medida a las empresas de productos que dependen sólo de esta fuente de ingresos. Como consecuencia, es difícil encontrar una empresa de productos pura de este tipo, ya que la seguridad de sus ingresos será demasiado discrecional y arriesgada y pasará de forma inevitable por periodos bajos.

Aunque cualquier actividad económica se verá afectada en este tipo de escenarios, las empresas centradas en servicios serán más capaces de mantener sus ingresos, dados los contratos a largo plazo con los que puedan contar, y dado que sus clientes serán principalmente otras empresas, que, aunque sea en menor medida, seguirán necesitando el mantenimiento de sus infraestructuras informáticas. En muchas ocasiones, estas infraestructuras permiten a la

empresa cliente funcionar con mayor eficiencia, aumentando así sus posibilidades de supervivencia en periodos bajos, por lo que mantendrán los gastos destinados a los servicios en torno a las nuevas tecnologías.

1.3. Modelos híbridos

En realidad, existen multitud de modelos híbridos, que combinan en distinto grado la venta de productos estándar, y la prestación de servicios, intentando compatibilizar ambas tendencias. Se puede considerar que el grado en el que una empresa se incline hacia los productos o servicios es indicativo de su propio ciclo de vida, y existe una tendencia generalizada de transición hacia los servicios.

Ejemplo de empresa de modelo híbrido

Pensemos en una empresa que empieza con un modelo puro de producto, logrando unas elevadas ventas, y unos grandes beneficios, pero que comprueba que será difícil mantener ese nivel de ingresos. Para asegurarse la continuidad, o como respuesta a periodos económicos bajos, podrá empezar a establecer contratos de servicios con algunos de sus clientes, viendo una reducción considerable en el ritmo de crecimiento de la empresa, pero obteniendo mayor estabilidad a largo plazo. Finalmente, puede que la empresa acabe por poner todo su peso sobre los servicios, habiendo saturado ya el mercado de su producto original.

Por supuesto, éste no es más que un ejemplo teórico, y muchas empresas ni completarán este ciclo, ni lo empezarán en el mismo sitio.

Por otro lado, la transición hacia los servicios no es fácil, y puede acarrear consecuencias negativas si no se hace con cuidado. Adoptar un modelo híbrido como respuesta a un momento de crisis, sin considerar detenidamente su estrategia de negocio, puede traer a una empresa de productos muchos problemas.

En momentos de falta de ingresos, la empresa puede ceder a presiones de distintos clientes para desarrollar adaptaciones del producto muy concretas, y de difícil integración en el producto estándar principal. Si esta práctica se generaliza, y la empresa pretende mantener sus ingresos mediante la venta del producto estándar, puede encontrar serias dificultades para mantener la compatibilidad entre las nuevas versiones lanzadas y las adaptaciones particulares de distintos clientes. El trabajo de depuración y desarrollo se multiplica, y en ocasiones puede llevar a la empresa a generar más gastos que ingresos.

1.4. Software como servicio

El concepto de "*software as a service*" (SaaS), o "**software como servicio**", se originó en 1999, como una nueva manera de implementación del software enfocada en la funcionalidad.

La visión básica de esta idea se centra en que, para los usuarios, el software se hace importante en la medida en que les permite solucionar un problema, es decir, en la medida en que les presta un servicio.

La necesidad de adquirir un producto de software, de contar con una infraestructura de hardware y software relacionado, y el trabajo necesario de instalación y soporte que conlleva, no serían, bajo este paradigma, más que un estorbo para el usuario final, que debe sobrellevarlas para poder obtener la funcionalidad deseada.

Bajo un modelo de software como servicio, todos estos inconvenientes desaparecen, y el software pasa de ser un producto que puede ser adquirido, a ser un servicio que puede ser prestado. En este sentido, es importante distinguir entre las empresas de servicios que hemos comentado anteriormente, que se dedican a **prestar servicios sobre un software** (instalación, mantenimiento, etc.), a este nuevo modelo que se enfoca sobre la **prestación del software como un servicio** (prestación de la funcionalidad concreta de ese software).

Para llevar a cabo esta idea, la empresa proveedora se haría cargo de toda la infraestructura necesaria, alojando el software requerido, y ofreciendo el servicio vía web a través de un navegador. La presencia de una infraestructura de comunicaciones suficientemente potente es necesaria, pero el resto de requerimientos tecnológicos en el lado del receptor del servicio disminuyen, permitiéndole centrar su atención sólo en la funcionalidad ofrecida.

Prestación de software como un servicio

Cada vez más empresas están usando este modelo para proporcionar software corporativo, como 37signals con Basecamp (herramienta de gestión de proyectos), y el popular Salesforce.com (CRM, *customer relationship management*) que permite la personalización del software en función de las necesidades de los clientes.

El modelo de *software como servicio* supone una opción de bajo coste para proporcionar software a empresas, frente a la venta de productos tradicional. Por un lado, los clientes se ahorrarán una cantidad importante en concepto de mantenimiento de sus infraestructuras tecnológicas. Por otra parte, las empresas proveedoras podrán ofrecer precios más económicos, al combinar los ingresos recurrentes derivados de la prestación de un servicio y aprovechar al mismo tiempo una única instancia de su aplicación para dar servicio a un gran número de clientes.

Software vía web

También encontramos numerosos ejemplos de software vía web orientado a consumidores individuales, aunque esta tendencia se ha denominado "Web 2.0". Muchas han tenido un gran éxito, como las numerosas aplicaciones de Google, o e-bay.

La presencia tanto de software libre como de ofertas SaaS está amenazando a los vendedores de software tradicionales, que están sintiendo una fuerte presión con la entrada de estos nuevos competidores, y tendrán dificultades a la hora de mantener los precios de sus productos.

Los proveedores de software como servicio, además, tienen mucho que ganar del empleo de software libre. Por un lado, al usarlo en sus infraestructuras de software conseguirán un ahorro considerable de costes en licencias o en desarrollo. Por otro lado, algunas están aprovechando aplicaciones libres, con licencia GPL, como base para desarrollar sus aplicaciones críticas de negocio, manteniendo sus modificaciones cerradas, como forma de proteger la diferen-

ciación de su negocio. Aprovechan así un vacío que contiene la GPL en este sentido: las modificaciones del código sólo deben ser redistribuidas si se redistribuye el programa. En el caso del software como servicio, el código no se llega a redistribuir, sólo lo hace la funcionalidad, por lo que la empresa no tiene la obligación de compartir sus mejoras.

2. Empresas dominantes en el sector

Como hemos visto, orientar una empresa hacia productos o servicios generará dinámicas empresariales muy distintas, pero ambas aproximaciones pueden dar lugar a modelos de negocio lucrativos. Sin embargo, mantener activas empresas de productos puras será muy difícil, y las barreras de entrada, enormes.

En la encuesta "Software 500" del "Software Magazine" (www.softwaremag.com), que cada año elabora un ranking de las primeras 500 empresas de software comercial según ingresos, podemos ver que entre las compañías más lucrativas se encuentra una representación de los dos tipos de empresas comentadas.

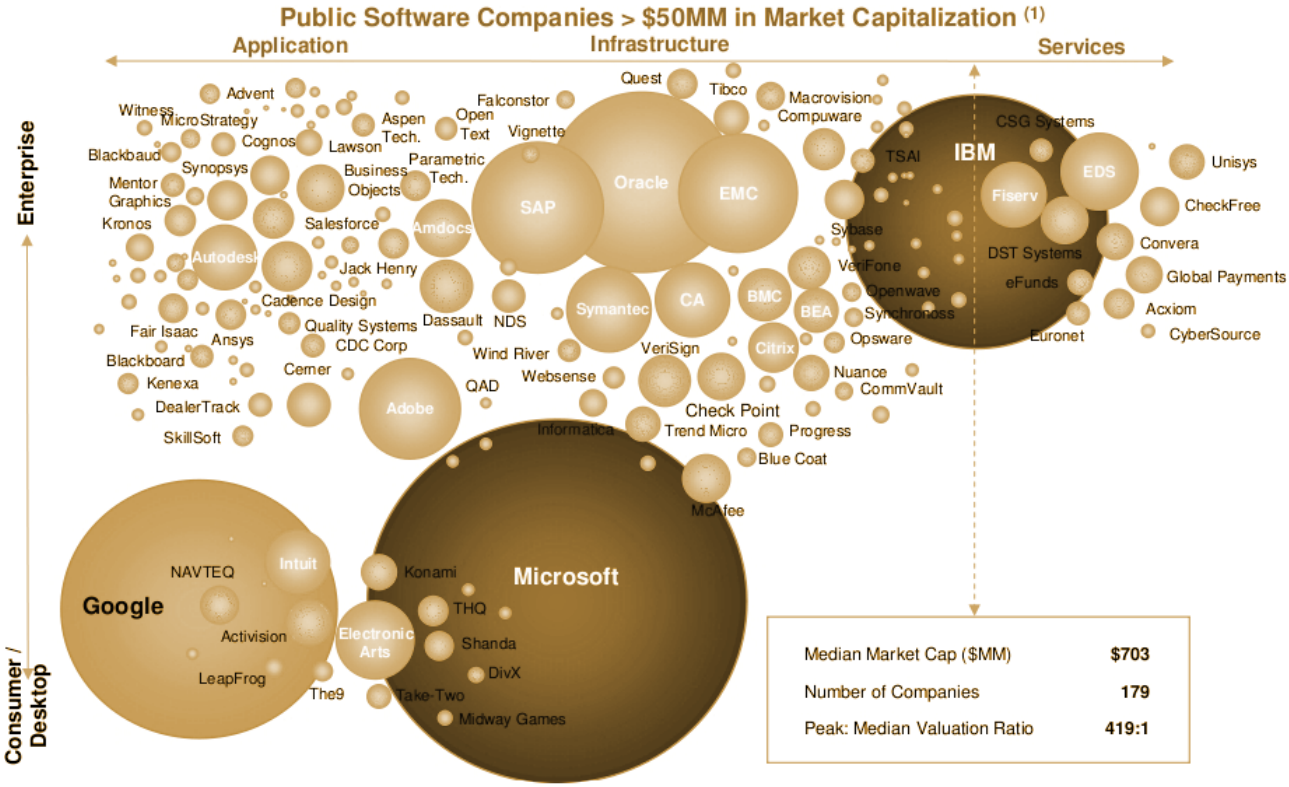
Sin embargo, de las veinte primeras, sólo cuatro de ellas mantienen un enfoque de producto marcado, con menos del 30% de servicios como porcentaje total de su actividad: Microsoft Corporation, Oracle, SAP y Symantec, representando productos líderes en sus sectores, orientados a clientes corporativos y a mercados de masas (sistemas operativos de escritorios, bases de datos, ERP y seguridad, respectivamente).

Dos empresas presentan un balance al 50% entre productos y servicios, Lockheed Martin Corporation y EMC Corporation. De las restantes, diez declaran como su sector de negocio principal la integración, la consultoría, y los servicios de subcontratación, y el resto, a pesar de dedicarse al desarrollo de productos concretos, obtienen sus ingresos principalmente de la prestación de servicios relacionados.

	Empresa	Sitio web	Ingresos derivados de software/ servicios (millones $)	Crecimiento de ingresos (%)	Servicios como %	Nº de personas empleada	Sector de software
1	IBM	www.ibm.com	$66,451.00	3.0%	72.6%	394,540	Middleware/Servidor de aplicaciones/Servidor web
2	Microsoft Corporation	www.microsoft.com	$39,317.00	9.0%	NA	71,000	Sistemas operativos
3	EDS	www.eds.com	$21,268.00	8.0%	100%	118,500	Servicios de outsourcing
4	Hewlett-Packard Company	www.hp.com	$16,918.00	2.0%	92.3%	156,000	Servicios de integración de sistemas/Consultoría IT
5	Accenture	www.accenture.com	$16,646.40	7.0%	100.%	140,000	Servicios de integración de sistemas/Consultoría IT
6	Computer Sciences Corporation	www.csc.com	$14,615.60	4.0%	NA	79,000	Servicios de integración de sistemas/Consultoría IT
7	Oracle Corporation	www.oracle.com	$14,380.00	22.0%	19.7%	56,133	Bases de datos
8	SAP	www.sap.com	$12,309.70	23.0%	29.2%	39,355	ERP (*enterprise resource planning*)
9	Cap Gemini	www.capgemini.com	$10,158.60	23.0%	NA	67,889	Servicios de integración de sistemas/Consultoría IT
10	Hitachi	www.hitachi.com	$9,019.20	5.0%	85.4%	356,000	*Storage management*
11	Lockheed Martin Corporation	www.lockheedmartin.com	$8,992.00	10.0%	51.2%	140,000	Aplicaciones verticales industriales
12	Science Applications International Corporation (SAIC)	www.saic.com	$7,775.00	8.0%	NA	43,600	Servicios de integración de sistemas/Consultoría IT
13	NTT Data Corporation	www.nttdata.co.jp	$6,685.80	4.0%	7.9%	21,308	Servicios de integración de sistemas/Consultoría IT
14	EMC Corporation	www.emc.com	$6,014.50	16.0%	51.2%	31,100	Gestión de la información
15	Affiliated Computer Services, Inc.	www.acs-inc.com	$5,353.70	23.0%	NA	58,000	Servicios de *outsourcing*
16	LogicaCMG plc	www.logicacmg.com	$5,221.40	65.0%	NA	40,483	Servicios de integración de sistemas/Consultoría IT
17	Unisys Corporation	www.unisys.com	$4,917.20	3.0%	NA	31,500	Servicios de integración de sistemas/Consultoría IT
18	Sun Microsystems, Inc.	www.sun.com	$4,697.00	19.0%	100.%	34,400	Middleware/Application Server/Web Server
19	SunGard Data Systems, Inc. Pvt	www.sungard.com	$4,212.00	8.0%	91.9%	16,600	Aplicaciones inancieras
20	Symantec Corporation	www.symantec.com	$4,143.40	60.0%	3.3%	17,396	Herramientas de seguridad/Sistemas

Primeras 20 empresas en el sector del software y su actividad principal (elaborada a partir del estudio "Software 500" de 2007. http://www.softwaremag.com/SW500/

En la siguiente figura podemos ver el posicionamiento actual de éstas y otras empresas de software, en cuanto a su enfoque (aplicación, infraestructura, servicios), y el tipo de clientes a los que se dirigen (empresas o consumidores individuales).



Posicionamiento de las principales empresas de software (con capitalización de mercado superior a $50 millones, y cotización en bolsa). John Prendergast (2008). "Can Xensource, MySQL or Jboss tell you anything about your company's prospects?". *Open Source Business Conference*. Disponible en: *http://http://akamai.infoworld.com/event/osbc/08/docs/CEO-CMO-Prendergast.pdf*)

3. Marketing en la empresa: ¿A quién vender?

Hasta ahora hemos examinado varios aspectos sobre la naturaleza principal de una empresa de software, y la definición de sus actividades principales. Sin embargo, otro aspecto fundamental que debe plantearse en cualquier empresa es qué vender, y a quién dirigirse.

3.1. Mercados de nicho y mercados de masas

Para cualquier empresa que cuente con unas fuertes economías de escala, como es el caso de las empresas de productos de software, cuanto mayor sea su base de usuarios, mayor será su margen de beneficios. Por lo tanto, la situación aparentemente más lucrativa sería la de orientar sus productos hacia los mercados de masas.

Sin embargo, una estrategia como ésta puede presentar muchas dificultades: el mercado de masas estará más estudiado, controlado y saturado por grandes empresas. Para una compañía que esté abriéndose camino, será extremadamente difícil llegar a competir con las ya establecidas y dominantes del sector, que además cuentan con una gran capacidad de marketing y difusión.

Resultará más fácil cubrir las necesidades detectadas en **mercados de nicho**, que por su tamaño no son atractivos para grandes empresas. Para las grandes compañías, los retornos potenciales de estos mercados son demasiado bajos, dado el reducido número de clientes, pero para una pequeña empresa serán más que suficientes. El número de nichos posibles es enorme, contando con numerosos factores sobre los que segmentar y concretar un mercado. La pregunta clave en este sentido será cuántos consumidores potenciales proporcionará este nicho, lo que permitirá calcular el volumen de negocio y por lo tanto, qué volumen de gastos podrá permitirse la empresa.

El mundo del software presenta posibilidades más interesantes que otros productos tangibles en los mercados de nicho, dada la ausencia de barreras geográficas que proporciona Internet. Un nicho detectado en un área geográfica dada podrá extrapolarse de manera relativamente fácil a otras zonas con necesidades similares, o incluso se ampliará solo, sin necesidad de un esfuerzo concreto por parte de la empresa comercializadora.

A la hora de crear productos para mercados de nicho, es fundamental conocer en profundidad ese entorno particular. Además de competencias técnicas, es necesario contar con un amplio conocimiento sobre las actividades, prioridades y forma de funcionamiento de ese nicho en particular. Siguiendo la regla de Eric Raymond, "todo buen trabajo de software comienza a partir de las necesidades personales del programador" ("*Every good work of software starts by scratching a developer's personal itch*") resulta útil partir de un nicho del que se forma parte, para entender mejor qué necesidades y problemas existen en él.

Conocimiento del entorno

Es el caso del nicho de los desarrolladores de software: está bastante explorado y explotado, ya que todas las personas programadoras son a la vez creadoras y usuarias, contando con un conocimiento íntimo de las necesidades y problemas del sector.

Otro factor importante a tener en cuenta es si se va a vender el producto a entornos corporativos, a pequeñas empresas o a personas individuales.

Las empresas de servicios deberán orientarse hacia entornos corporativos, administraciones públicas u otras organizaciones, ya que las personas consumidoras raramente pagarán por servicios orientados al software. Las empresas de productos, sin embargo, podrán elegir, de acuerdo a las características de sus productos y a su estrategia empresarial, los clientes potenciales del mercado objetivo. Los clientes corporativos pueden resultar más atractivos, ya que estarán más dispuestos a pagar por un producto de software, y además contribuirán a generar ingresos a través de servicios.

En un entorno corporativo, se pagará por un producto de software, pero también por el soporte sobre ese producto, la formación, la instalación y la integración en el resto de sus sistemas. Las empresas que compran software, en general pagarán del 15% al 25% del precio de la licencia en concepto de mantenimiento anual (Dan Woods, Gautam Guliani, "Open source for the enterprise"). A menudo, también solicitarán desarrollos a medida para adaptar el producto a sus propias necesidades. De esta manera, el cliente corporativo ayudará a una empresa de software a desarrollar ingresos a partir de servicios, dándole más garantías de continuidad. Sin embargo, estos nuevos ingresos serán más intensivos en cuanto a mano de obra, y será necesario llevar una gestión de la empresa cuidadosa para asegurar que los costes de la prestación del servicio no superan los ingresos derivados de ellos.

Lectura recomendada

D. Woods; G. Guliani (2005). *Open source for the enterprise: managing risks, reaping rewards*. O'Reilly Media, Inc.

Por otro lado, los servicios de soporte a menudo se ofrecen sobre versiones concretas del producto, por lo que mantener una relación de servicios también puede ayudar a generar ingresos en forma de licencias de sucesivas versiones: aunque el cliente no tuviera interés por adquirir la nueva versión, se verá obligado dado que el soporte sobre la versión antigua ya no se realiza.

Como desventaja, un cliente corporativo de importancia será reacio a contratar servicios de una pequeña y nueva empresa. Uno de los factores esenciales a la hora de la contratación será la reputación y la confianza que le genere la compañía proveedora de servicios, por lo que las empresas menores o de reciente creación encontrarán clientes más fácilmente en su mismo entorno, es decir, entre pequeñas y medianas empresas.

3.2. Patrones de adopción tecnológica y el "abismo"

Detectar un nicho de mercado y elaborar un buen producto que satisfaga las necesidades del grupo de usuarios potenciales no es suficiente para lograr su aceptación. Para lograr introducir un nuevo producto o servicio, será fundamental tener en cuenta cómo son los patrones de adopción de tecnología entre un grupo de personas.

Los libros de marketing tradicionalmente han dibujado un modelo de adopción sobre la base de una curva de Gauss con cuatro grupos de usuarios:

- **Innovadores y primeros en probar (*early adopters*):** les gusta la tecnología y la innovación. A menudo adoptarán cierto producto sólo porque es nuevo.

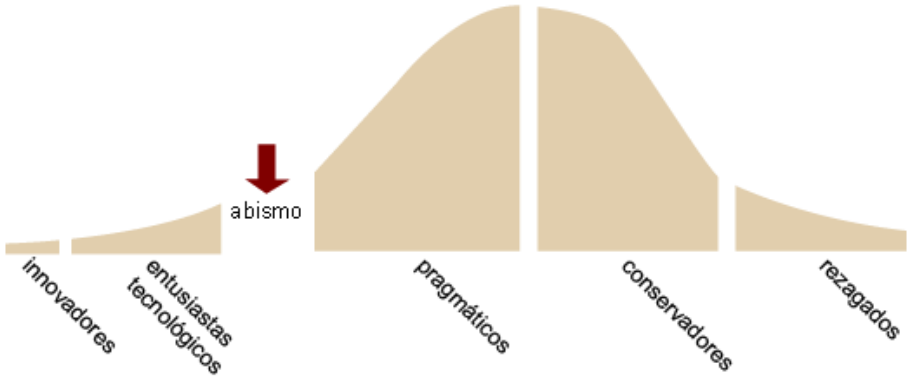
- **Mayorías tempranas:** adoptarán una tecnología sólo si les ayuda a solucionar un problema concreto.

- **Mayorías tardías:** intentan evitar las nuevas tecnologías.

- **Rezagados:** serán los últimos en probar algo nuevo, o puede que nunca lleguen a probarlo.

La curva representa dos ideas clave: las dos categorías intermedias agrupan a la gran mayoría de los **clientes potenciales**, y sólo se puede conseguir captar a los grupos en orden, de izquierda a derecha (los "primeros en probar" la adoptarán si lo han hecho los innovadores, las mayorías tempranas, si lo han hecho los innovadores, las mayorías tardías, si lo han hecho las tempranas, y los rezagados, si lo han hecho las mayorías tardías).

Geoffrey Moore, en su libro *Crossing the Chasm* renombra los grupos, refiriéndose a ellos como **entusiastas tecnológicos**, **visionarios**, **pragmáticos**, **conservadores** y **escépticos**, y argumenta que esta teoría tiene un fallo, ya que la transición entre los entusiastas y la mayorías pragmáticas no es un continuo, y será difícil de conseguir. Las mayorías tempranas no adoptarán soluciones que no hayan sido ampliamente probadas, y lo harán de las que puedan obtener buenas referencias de otros pragmáticos, por lo que a veces alcanzarlas puede parecer una tarea imposible. Para Moore, existe un abismo (*chasm*) entre ambos grupos, por lo que redibujó la curva de la siguiente manera:



Curva de adopción tecnológica según Moore

Los innovadores y los entusiastas tecnológicos tienen una **alta tolerancia al riesgo** y a los defectos de la nueva tecnología, ya que cuentan con una importante habilidad técnica. Estos usuarios adoptarán una tecnología concreta sobre la base de la funcionalidad pura que presenten buscando la innovación. Las mayorías tempranas y tardías (pragmáticos y conservadores) tienen una **tolerancia baja al riesgo**, y estarán interesadas en adquirir un producto que les permita aumentar su productividad, pero que cuente con una alta estabilidad y madurez.

De esta manera, un producto innovador podrá tener un éxito importante entre los innovadores y los entusiastas tecnológicos, pero si la compañía creadora quiere ampliar su base de clientes, tendrá que poner en marcha una campaña de marketing distinta, poniendo énfasis no en las funcionalidades concretas y mejoras que ofrece el producto, sino en generar confianza sobre él, contando casos de éxito, implementaciones previas y número de usuarios.

Conseguir los primeros clientes en el grupo de los pragmáticos y mantenerlos contentos se hace fundamental, pero resulta muy difícil, dado el círculo vicioso que se crea: ninguno de ellos adoptará una solución que otros pragmáticos no hayan probado antes.

Se puede construir confianza ofreciendo soluciones completas, que incluyan mantenimiento, soporte y formación, para lograr atraer a clientes sensibles a la estabilidad y facilidad de uso del producto. Los primeros clientes de este grupo tendrán que ser tratados con cuidado, sin escatimar en tiempo o coste, ya que serán el punto de referencia para el resto. Una vez conseguidos unos pocos pragmáticos de referencia, atraer al resto será mucho más fácil, y una vez los pragmáticos hayan adoptado la solución, los conservadores seguirán sin requerir esfuerzos de marketing especiales.

Centrarse en los innovadores y entusiastas –por suponer que a pesar de ser un mercado potencial reducido, será suficiente para una pequeña empresa– puede ser peligroso, ya que este grupo es inestable por naturaleza, y abandonará un producto en cuanto deje de ser novedoso.

Esta curva de adopción también marcará el ciclo de vida del producto, sus dinámicas de desarrollo y sus prácticas de marketing. La empresa comercializadora debe tener claro en qué fase está y quiénes son sus clientes en ese momento, ya que cada grupo se siente atraído por muy distintos factores. Mientras que incluir muchas funcionalidades nuevas, y mantener un producto cambiante atraerá a los innovadores, los conservadores necesitarán que el producto simplemente funcione en unos escenarios concretos, y que lo haga siempre igual. Cada cambio supondrá una dificultad que sólo estarán dispuestos a superar si ello conlleva la solución de algún problema con el que se encuentren.

4. Función del producto: ¿Qué vender?

Considerar con cuidado la naturaleza del producto a desarrollar es muy importante. Una de las preguntas que habrá que resolver es si se pretende que el producto sea líder en el sector, un seguidor o un producto complementario.

Aunque en principio ser el líder del sector puede parecer lo más atractivo, puede no ser lo más efectivo. Al detectar la ausencia de una funcionalidad en un producto con una amplia adopción, una empresa podría tomar dos caminos: desarrollar su propia versión, incluyendo la funcionalidad ausente e intentar competir con el líder, o construir un complemento al líder que complete sus posibilidades.

La primera opción resultará muy complicada, y puede fracasar fácilmente, necesitando una inversión considerable no sólo en el nuevo desarrollo, sino en la campaña de marketing y ventas posterior. En la segunda, además de poder desarrollar el producto en menos tiempo, el trabajo de marketing estará ya hecho, en gran medida, por el líder, por lo que será mucho más fácil que el complemento gane adopción. Además, los usuarios conservadores (las mayorías) estarán mucho más dispuestos a incorporar un complemento a una solución conocida y probada, que a cambiar de tecnología y de proveedor. Un peligro común será que la empresa líder decida incorporar la funcionalidad desarrollada a su producto central, eliminando la necesidad de adquirir el complemento. En este sentido, la relación que se tenga con la empresa desarrolladora del producto central será fundamental.

Resulta importante, por lo tanto, definir el papel que tendrán otras empresas activas en el sector. Cuáles van a actuar como competidoras directas, cuáles como colaboradoras y cuáles, aun estando en el mismo sector, no entrarán a competir con nuestro producto por contar con una especialización concreta. Segmentando nichos y ofreciendo diferenciación, se puede evitar la competencia directa de empresas fuertes, y la existencia de empresas que elaboren productos o servicios relacionados puede ser un factor importante para el éxito.

A la hora de posicionar un producto, también será importante tener en cuenta para qué plataforma se va a desarrollar, es decir, qué conjunto de software básico es necesario para el funcionamiento del producto. Pensemos, por ejemplo, en la elección del sistema operativo y tecnología relacionada con el que operará la aplicación. Esta decisión afectará a la definición del nicho de mercado que se va a explotar, y a qué tipo de clientes se podrá dirigir, pero también será importante a la hora de definir su relación con aliados y competencia.

Una aplicación diseñada para funcionar en una plataforma determinada será una aplicación complementaria de dicha plataforma. Si se trata de un conjunto de software ya consolidado en el mercado y de amplia aceptación, se amplía también el mercado potencial de clientes, pero se reducen las posibilidades de encontrar aliados en los desarrolladores de la plataforma. El valor de estas plataformas vendrá dado, en gran medida, por el número y diversidad de aplicaciones que corran sobre ella, por lo que una empresa que esté intentando establecerse como líder de plataforma tendrá bastante interés en que se desarrollen aplicaciones relacionadas, y será, por lo tanto, una aliada más dispuesta.

Sin embargo, aunque más difícil, puede ser más interesante posicionarse como líder en un sector determinado. La pregunta, en este caso, será si se pretende crear una categoría de producto nueva para un nicho sin explotar, o si se va a intentar desplazar a otro producto existente.

La segmentación y los clientes potenciales

Para una empresa modesta, la única posibilidad puede ser la de ir segmentando el mercado, hasta encontrar un nicho concreto donde posicionarse. Puede ser difícil posicionarse como líder en las aplicaciones de planificación de recursos empresariales (ERP, *enterprise resource planning*), pero puede ser más fácil si se desarrolla un ERP para pymes, o un ERP para pymes hosteleras. Desde luego, al aumentar la segmentación, disminuirá la competencia, aunque también la base de clientes potenciales.

Ser los primeros en un mercado determinado sin duda proporcionará ventajas a la hora de posicionarse como líder, y a la hora de definir los estándares sobre los que se asentará esa tecnología, pero no proporciona ninguna garantía. La primera empresa en desarrollar una tecnología no siempre llega a convertirse en líder en el sector. A veces, llegar primeros y capturar a los entusiastas tecnológicos puede dar falsas indicaciones de éxito, ya que el producto deberá alcanzar a las mayorías para convertirse en líder. Las decisiones estratégicas y técnicas posteriores serán decisivas para determinar si la empresa es capaz de capitalizar las economías de escala de la demanda para posicionar a su producto como número uno en su sector.

Para conseguir irrumpir en un mercado que ya cuenta con un líder, harán falta campañas de marketing y ventas que a menudo no están al alcance de empresas de reciente creación. Sin embargo, el uso de un producto de software libre, que entre a competir con precio cero, puede ser un agente disruptor suficientemente potente. En próximos módulos veremos ésta y otras estrategias que proporciona el software libre para competir en distintos mercados.

Resumen

Las necesidades en torno al software generan múltiples oportunidades de negocio a lo largo de su ciclo de vida, desde el propio desarrollo hasta servicios conexos como la instalación, la migración o la formación de los usuarios.

El posicionamiento empresarial resulta clave para determinar las posibilidades del negocio:

- La orientación a servicios proporciona un marco económico más estable al cabo del tiempo.

- La orientación al desarrollo de productos genera una economía de producto, más difícil de mantener en periodos largos.

- Los modelos híbridos pretenden ofrecer garantías de equilibrio a los dos modelos anteriores.

- La irrupción del software como servicio amenaza los modelos más tradicionales, ofreciendo una variación más versátil para los clientes potenciales.

Por otra parte, la explotación de vetas de mercado que son próximas y conocidas puede ayudar a la estrategia empresarial de un nuevo negocio, así como la adecuación del producto a los patrones de adopción tecnológica del mercado objetivo.

Finalmente, también será necesario establecer claramente la relación del negocio con sus competidores, así como la del producto con la de sus competidores. Estas relaciones pueden eventualmente favorecer la introducción del producto en el mercado objetivo.

Bibliografía

Christensen, C. M. (1997). *The innovator's dilemma*. Harvard University Press <http://books.google.es/books?id=SIexi_qgq2gC> [Fecha de consulta: abril 2009]

Christensen, C. M.; Raynor, M. E. (2003). *The innovator's solution*. Harvard University Press. <http://books.google.es/books?id=ZUsn9uIgkAUC> [Fecha de consulta: abril 2009]

Cusumano, M. (2004). *The Business of Software* Free Press. Cambridge: Cambridge University Press. <http://books.google.com/books?id=7KAWToDnBAC&dq=the+business+of+software&hl=es> [Consulta: febrero 2009]

Daffara, C. (marzo, 2006). "Sustainability of FLOSS-based economic models". *II Open Source World Conference*. Malaga. <http://www.cospa-project.org/Assets/resources/daffara-OSWC2.pdf> [Fecha de consulta: abril 2009]

McKenna, R.; Moore, G. (2006). *Crossing the chasm* Capstone. <http://books.google.com/books?id=GTwFAQAACAAJ&dq=crossing+the+chasm&hl=es> [Consulta: febrero 2009]

Sink, E. (2006). *Eric Sink on the Business of Software* Apress. New Jersey: Princeton University Press <http://books.google.com/books?id=h5IQuengOGIC&dq=eric+sink+business+of+software> [Consulta: febrero 2009]

Modelos de negocio con software libre

Irene Fernández Monsalve

PID_00145049

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

En los módulos anteriores hemos examinado el mercado del software, los tipos tradicionales de empresas en el sector, y las posibilidades que brinda el software libre dentro de este marco. Dedicaremos este módulo al estudio de los modelos de negocio más usuales que se crean en torno al software libre, y de algunos casos concretos.

Sin ninguna duda, el software libre está emergiendo en los últimos años como un elemento clave en nuevos modelos de negocio. Después del desinfle de la burbuja tecnológica (popularmente conocida como la burbuja de las punto com) a principios de esta década, el software libre ha supuesto un motor para la creación de nuevas empresas en el sector tecnológico, consiguiendo atraer crecientes cantidades de capital-riesgo. En el 2004, se invirtió un total de $149 millones, distribuidos entre 20 nuevas empresas. En el 2006, la cantidad subió a $475 millones, distribuidos entre 48 iniciativas empresariales.

Bibliografía recomendada

Para más información podéis consultar: Larry Augustin (2007). "A New Breed of P&L: the Open Source Business Financial Model". *Open Source Business Conference (OSBC)*.

http://www.osbc.com/live/images/13/presentation_dwn/A_New_Breed_of_P_and_L.pdf

A las empresas ya establecidas y reconocidas, como Red Hat o MySQL, se ha unido una nueva generación de numerosas empresas que centran sus estrategias en el uso y desarrollo de software libre. En los próximos años veremos cuál es la evolución real de estas nuevas empresas, y si su modelo de negocio sostenible a largo plazo.

Primera generación	Segunda generación	Tercera generación
Con cotización en bolsa: Red Hat, Caldera (ahora SCO), VA Linux (ahora VA Software), Turbolinux **Adquiridas**: SUSE, Cygnus **Otras**: LynuxWorks, Linuxcare (ahora Levanta), Sendmail	**Con cotización en bolsa**: Trolltech, Sourcefire, Mandrakesoft (ahora Mandriva) **Adquiridas**: Conectiva, Lycoris, JBoss, Sleepycat, Ximian, Gluecode **Otras**: MontaVista, MySQL, Zend	ActiveGrid, ActiveState, Alfresco, BitRock, Black Duck, CollabNet, Collax, Compiere, Covalent, DB4O, Digium, Exadel, eZ Systems, Fonality, Funambol, Groundwork, Hyperic, Ingres, Interface21, JasperSoft, Joomla, Laszlo Systems, Medsphere, Mozilla Corp, MuleSource, OpenBravo, OpenLogic, OpenXchange, OTRS, Palamida, Pentaho, rPath, SnapLogic, Sourcelabs, Spikesource, SQLite, WebYog, SugarCRM, Talend, Terracotta, Ubuntu / Canonical, Vyatta, WSO2, XenSource, Zenoss, Zimbra, Zmanda, etc.

Modelos de negocio con software libre: casos de éxito. Tabla y datos de inversiones obtenidos de Marten Mickos (2007). "Open Source: why freedom makes a better business model". *Open Source Business Conference (OSBC)*. http://www.osbc.com/live/images/13/presentation_dwn/Keynote-Open_Source_Why_Freedom.pdf)

En este módulo vamos a estudiar algunos de los casos empresariales que se presentan en la tabla anterior, además de otros que, si bien no atraen la atención por sus reducidas dimensiones, resultan bastante significativos. Nos detendremos en las ventajas del software libre que explotan, los problemas que encuentran, y cómo los solucionan. Además, examinaremos distintas taxonomías para caracterizar estos modelos, intentando identificar distintos factores clave que determinan el funcionamiento de la empresa según distintos autores.

Objetivos

Al finalizar este módulo, se deben alcanzar los objetivos siguientes:

1. Entender las principales clasificaciones de modelos de software libre que se han confeccionado hasta ahora.

2. Conocer los modelos de negocio basados en software libre existentes en la actualidad.

3. Entender los distintos mecanismos de generación de ingresos, y de diferenciación que explotan dichos modelos.

4. Ser capaz de analizar cómo aprovechan el software libre distintas empresas para generar una ventaja competitiva.

1. Caracterizando modelos de negocio con software libre

Cuando hablamos de modelos de negocio basados en software libre, a menudo nos referimos a qué nuevas e ingeniosas maneras de obtener ingresos son puestas en marcha, ya que el modelo tradicional, de venta de un producto propietario, ya no es evidente. Las empresas, a diferencia de las personas individuales, deben considerar un importante factor a la hora de participar en un proyecto de software libre: cómo capturar un retorno económico que justifique su inversión.

Tal como hemos visto en módulos anteriores, la idea de que los ingresos que genera el software están directamente vinculados a su venta no se ajusta totalmente a la realidad. La mayor parte del software se desarrolla internamente, y son pocas las empresas para las que la venta de software supone el principal ingreso. En la mayoría de los casos, la oferta de servicios complementarios se hace necesaria para asegurar la continuidad de los ingresos y la supervivencia de la empresa en momentos difíciles.

Por otra parte, en el artículo de Perens examinado en el segundo módulo ("The Emerging Economic Paradigm of Open Source"), hemos visto que el software libre ofrece unas perspectivas económicas (coste y riesgo) mucho mejores que las alternativas propietarias para empresas que necesiten desarrollar software no diferenciador.

Web recomendada

Para más información sobre Perens:

http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1470/1385

En cualquier caso, en este módulo veremos cómo distintas empresas gestionan la propiedad intelectual de sus productos, generando también **modelos mixtos**, que tratan de compatibilizar las ventajas de los modelos libres con la captura de retornos económicos directos en base a la propiedad intelectual. En este sentido, la elección de licencia determinará, en gran medida, el rango de modelos de negocio que una empresa pueda implementar.

2. Clasificaciones según distintos autores

En este apartado, vamos a dar un repaso a distintos intentos de clasificación de modelos de negocio que aparecen en la literatura, deteniéndonos en los factores que cada autor ha considerado claves para agrupar distintos modelos. Además de aproximaciones más teóricas, veremos otras que se han basado en observar empresas existentes de una forma más cualitativa, así como una metodología cuantitativa de clasificación de modelos de negocio en el contexto del **proyecto FLOSSmetrics**. Finalmente, propondremos una taxonomía propia que de alguna manera aúne las propuestas examinadas.

2.1. Clasificaciones de Hecker y Raymond

Uno de los primeros autores en escribir sobre las perspectivas de negocio que ofrecía el software libre fue Frank Hecker en 1998, en "Setting Up Shop: The Business of Open-Source Software". En su artículo, toma cuatro categorías de OpenSource.org, y añade otras más, examinándolas en base a:

Web recomendada

Para más información podéis consultar:
http://hecker.org/writings/setting-up-shop

- Qué empresas ponen en práctica dicho modelo.
- Qué tipos de licencias son apropiadas.
- Qué oportunidades de diferenciación presenta el modelo.
- Qué oportunidades proporciona el modelo para poner precios sobre la base del valor percibido en lugar de sobre la base de costes reales.

A continuación presentamos una tabla resumen de su clasificación, añadiendo otro parámetro de caracterización que, aunque no es mencionado explícitamente por Hecker, constituye una característica fundamental: cómo se originan los ingresos de la empresa.

Modelo	Fuente de ingresos	Tipo de licencia	Oportunidades de diferenciación	Oportunidades de precios basados en valor percibido *vs.* costes	Casos
Support sellers	Venta de servicios relacionados (agrupa todo tipo de servicios, desde desarrollos a medida, a formación, consultoría, etc.)	GPL	Calidad, precio, y simplificación y mejora de la experiencia de usuario.	Limitada. Posible si goza de mucha reputación.	Cygnus Solutions Red Hat Caldera
Loss leader	Venta de otros productos propietarios	BSD o Mozilla	Basada en el producto.	Posible.	Sendmail Netscape

Resumen de clasificación de modelos de negocio ("Setting up shop: the business of open source" Hecker, 1998)

Modelo	Fuente de ingresos	Tipo de licencia	Oportunidades de diferenciación	Oportunidades de precios basados en valor percibido *vs.* costes	Casos
Widget frosting	Venta de hardware		Basada en el hardware – funcionalidad, rendimiento, flexibilidad, fiabilidad, coste...	Limitada. Típicamente el esquema de precios de hardware está basado en costes.	Corel VA Linux
Accessorizing	Venta de productos físicos (libros, etc.)		Calidad del producto (libros, etc.), y lealtad por parte de usuarios "pro-software libre".	Limitada. La reputación de marca puede permitir elevar un poco los precios.	O'Reilly & Associates
Service enabler	Venta de servicios en línea proporcionados por el programa	GPL o Mozilla	Atributos del *backend*, creación de servicios únicos y útiles.	Posible en la medida que se logra crear un servicio único y no imitable.	Netscape
Sell it, free it	Como "loss leader" cíclico	BSD o Mozilla	Funcionalidad del software (mientras se mantiene cerrado).	Posible, hasta que el producto se convierte en un bien intercambiable (en ese momento se libera)	–hipotético–
Brand licensing	Venta de derechos de marca. Convive la versión con marca con la versión "genérica".		Valor añadido por ejemplo, mediante validación y testeo adicional al producto sin marca.		–hipotético–
Software franchising	Venta de franquicia y porcentaje sobre ingresos de las franquicias		Como *support-seller y brand licensing*	Posible si se goza de reputación.	–hipotético–
Híbridos (licencias no son ni libres ni propietarias puras)	Limitar disponibilidad de código: venta de licencias bajo condiciones determinadas				Trolltech Qt
	Tratamiento según usuarios – venta a usuarios comerciales				Open Group
	Tratamiento según uso – venta para usos comerciales, o venta para uso en determinadas plataformas				Qt

Resumen de clasificación de modelos de negocio ("Setting up shop: the business of open source" Hecker, 1998)

En *El Caldero Mágico*, Eric S. Raymond también dibuja el papel del software libre en el mundo empresarial, centrándose, entre otros, en cómo el software libre afecta al "**valor de uso**" (**valor como producto intermedio)** y al "**valor de venta**" (**valor como producto final**) del software, proponiendo una taxonomía basada en cuál de los dos explota la compañía.

Web recomendada

Para más información podéis consultar:
http://catb.org/~esr/writings/magic-cauldron/

Para Raymond, sólo el valor de venta se ve afectado por un modelo de software libre, por lo que su clasificación describe modelos basados en valor de uso, y modelos basados en el valor de venta indirecto, en los que el software libre hace viable la venta de otro producto o servicio:

- Modelos basados en el valor de uso
  - Compartir costes (por ejemplo, Apache)
  - Compartir riesgo (por ejemplo, Cisco)

- Modelos basados en el valor de venta indirecto
  - Posicionador de mercado (*loss-leader/market positioner*)
  - Venta de hardware (*widget frosting*)

- Regala la receta, abre un restaurante (*give away the recipe, open a restaurant*)
- Venta de accesorios (*accessorizing*)
- Libera el futuro, vende el presente (*free the future, sell the present*)
- Libera el software, vende la marca (*free the software, sell the brand*)
- Libera el software, vende el contenido (*free the software, sell the content*)

Como podemos ver, Raymond incluye entre los modelos basados en el valor de venta indirecto los recogidos por Hecker, más uno nuevo "Free the software, sell the content". En este modelo, el valor está en la información proporcionada por la plataforma de software, y es esa información la que se vende mediante subscripciones. El software se libera, consiguiendo que sea portado a distintas plataformas, ampliando por lo tanto el mercado potencial del verdadero producto: **el contenido**.

Aunque se propone sólo como modelo hipotético, Raymond se adelanta a los conceptos de "**web social**", y de cambio de paradigma que propone O'Reilly en su artículo "Open Source Paradigm Shift".

Web recomendada

Más información sobre "Open Source Paradigm Shift en:

http://www.oreillynet.com/pub/a/oreilly/tim/articles/paradigmshift_0504.html

Sin embargo, no llega a reconocer el papel de Internet como plataforma, y del posterior *software as a service*, creyendo que el valor de liberar el software será el de conseguir que se porte a otras plataformas, contribuyendo a la difusión y a la ampliación del mercado.

2.2. Clasificación del Grupo de Trabajo Europeo de Software Libre (European Working Group on Libre Software)

Los modelos de negocio presentados por Hecker y Raymond se basan en la observación de empresas que usaban software libre como parte de sus modelos de negocio, pero quizás les falta cierta sistematización y abstracción en su taxonomía. El **grupo de trabajo europeo** de **software libre**, en su documento "Free Software/Open Source: Information Society Opportunities for Europe?" (http://eu.conecta.it/paper/), hace su análisis sobre la base de cómo se financian los proyectos de software libre en lugar de hacerlo sobre la base de los modelos de negocio, e independientemente de si el proyecto tiene relación con alguna empresa en concreto o no:

- Financiación pública.
- Financiación privada sin ánimo de lucro.
- Financiación por quien necesita mejoras.
- Financiación con beneficios relacionados (O'Reilly y Perl).
- Financiación como inversión interna.

- Otros (bonos, cooperativas de desarrolladores, utilización de mercados para poner en contacto clientes y desarrolladores).

Dentro de su apartado "Financiación como inversión interna", sin embargo, encontramos una clasificación de modelos de negocio, que destaca, entre otras, la posibilidad de generar **ingresos mediante servicios**, gracias a la ventaja competitiva que otorgaría ser los principales desarrolladores de un proyecto de software dado.

Modelo	Diferenciación	Ingresos	Licencias	Ejemplos
Mejor conocimiento	Mejor conocimiento del producto: debe ser la desarrolladora del producto, o una colaboradora.	Servicios relacionados: desarrollos a medida, adaptaciones, instalación, integración.	Libres	LinuxCare (en sus inicios) Alcove
Mejor conocimiento con limitaciones	Mejor conocimiento del producto: debe ser la desarrolladora del producto. Se mantiene una parte propietaria.	Servicios relacionados, y venta de parte propietaria.	Libres y propietarias	Caldera Ximian
Fuente de un producto libre	Productora, casi en su totalidad de producto libre.	Servicios relacionados: desarrollos a medida, adaptaciones, instalación, integración.	Libres	Ximian Zope Corporation
Fuente de un producto libre con limitaciones	Producto propietario en principio. Liberación posterior como estrategia para ganar adopción y otras ventajas del software libre.	Venta de versión comercial.	Libres y propietarias	Artofcode LLC Ada Core Technologies
Licencias especiales	Mejor conocimiento. Oferta de versión propietaria para clientes que no quieran GPL.	Venta de versión comercial, y servicios relacionados.	GPL y propietarias	Sleepycat
Venta de marca	En base a imagen y marca, que permitan vender producto a mayor precio.	Venta de distribuciones, y servicios relacionados (incluyendo certificación y formación)	Libres	Red Hat

Modelos de negocio basados en software libre. (Manual didáctico de "Introducción al Software Libre")

2.3. Estudios empíricos

En "Business models in FLOSS-based companies", Carlo Daffara describe un estudio empírico de los modelos de negocio de empresas basadas en el uso de software libre, emprendido en el contexto del **proyecto FLOSSmetrics**. El estudio también examina cómo esos modelos manejan la comercialización de sus productos, y qué licencias emplean.

Web recomendada

Para más información podéis consultar:

http://opensource.mit.edu/papers/OSSEMP07-daffara.pdf

El estudio partió con 120 empresas, de las que se eliminaron aquellas que no se consideraron basadas en FLOSS (*free, libre and open source software*), como las que permitían acceso al código sólo a usuarios no comerciales o que no permitían la redistribución. También se eliminaron empresas que, a pesar de prestar contribuciones importantes a proyectos de software libre, no basan su modelo de negocio principal en él (como IBM, HP y SUN).

Se seleccionó un conjunto de aspectos caracterizadores, como licencia, productos y servicios ofrecidos (instalación, integración, formación, consultoría, certificaciones técnicas y legales), tipos de contratos (subscripciones, licencias, o por-incidencia), así como literatura auto-referencial ofrecida en sus páginas web, e información relativa a su relación con la comunidad. Finalmente, se recogieron los datos, y se eliminaron las variables no significativas, obteniendo las siguientes variables caracterizadoras:

- Principal generador de ingresos
  - Selección.
  - ITSC (*installation/training/support/consulting)*. Se agrupan los distintos tipos de servicios, ya que el estudio encuentra que las empresas que ofrecen uno de ellos tienden a ofrecer también el resto.
  - Subscripciones.
  - Licencias.

- Modelo de licencia

Aplicando un análisis de conglomerados (análisis de *clusters*) a las empresas caracterizadas en términos de estas variables, el estudio obtiene seis modelos principales de negocio, y un séptimo grupo analizado por separado:

1) **Doble licencia:** Esquema de doble licencia GPL y propietaria para vender a quienes quieran desarrollar código cerrado basado en el producto libre.
2) **Productos separados OSS y comerciales:** Venta de productos comerciales basados en otro libre.
3) "**Badgeware**"**:** Protección de marca, productos liberados deben mantener logo/autoría original visible.
4) **Especialistas de producto:** Creación de un producto libre y comercialización de servicios entorno a él.
5) **Proveedores de plataformas:** Servicios de selección, integración y soporte, proporcionando plataformas probadas y verificadas.
6) **Empresas de selección/consultoras:** Analistas y servicios genéricos que, en genaral no contribuyen a la comunidad, dado que los resultados del análisis y consultoría se mantienen privados.
7) **Mercados auxiliares:** Como ejemplo, SourceForge/OSTG recibe la mayoría de sus ingresos de las ventas de su sitio afiliado ThinkGeek. Aunque este modelo no es uno de los caracterizados por el estudio (el limitado número de casos de esta categoría no permite la extrapolación), no debe subestimarse, suponiendo un modelo de financiación importante.

A continuación, mostramos la tabla de resultados obtenida del estudio.

		Main Licensing model					Main revenue generation			
	Company	twin licensing	OSS and commercial versions	Badgeware	Pure OSS	multiple packages covered	selection	ITSC	Subscription	licensing
twin lic.	Funambol	•						•		•
	Lustre	•						•		
	MuleSource	•							•	•
	Mysql	•							•	•
	OpenClovis	•							•	
	Pentaho	•						•		•
	sleepycatdb	•								•
Split OSS/ commercial releases	Adaptive Planning		•							•
	Alterpoint		•					•		•
	Altinity		•					•		•
	Codeweaver (WINE)		•							•
	Coupa		•							•
	Digium (Asterisk)		•						•	
	Enormalism		•							•
	EnterpriseDB		•							•
	GreenPlum		•							•
	GroundWork		•						•	
	Hyperic		•						•	
	JasperSoft		•							•
	Knowlede Tree		•	•						
	OpenCountry		•							•
	Open-Xchange		•							
	NoMachine NX		•							•
	rPath		•						•	
	Scalix		•							•
	Sendmail		•							•
	Smoothwall		•						•	
	Sourcefire (SNORT)		•						•	
	Splunk		•						•	
	SSLExplorer		•							•
	SugarCRM		•	•						•
	TenderSystem		•	•						•
	VirtualBox		•							•
	Vyalta		•					•	•	
	XenSource(Xen)		•					•		
	Zen(PHP)		•							•
	Zimbra		•	•						•
Badgeware	1bizcom			•				•		
	CATS applicant tracking			•					•	
	EmuSoftware/Netdirector			•				•	•	
	Jbilling			•				•		
	OpenBravo			•				•		
	OpenEMM			•				•		
	Open Terracotta			•					•	
	SocialText			•						•
product specialists	Alfresco				•			•	•	
	Babel				•			•		
	CentraView				•			•		
	CleverSafe				•			•		
	Compiere				•			•	•	
	Exadel				•			•		
	Jitterbit				•			•	•	
	Mergere				•			•		
	Mindquarry				•			•		
	Mirth				•			•		
	OfBIZ				•			•		
	Qlusters (OpenQRM)				•			•		
	Symbiot/OpenSIMS				•			•		
	Talend				•			•		
	UltimateEMR				•				•	
	VISTA				•			•		
	vTiger				•			•		
	Zenos s				•				•	
Platf. Prov.	Jbos s				•	•		•	•	
	RedHat Linux				•	•			•	
	Sourcelabs				•	•	•		•	
	SpikeSource				•	•	•		•	
	SUS Linux				•	•			•	
	WSO2				•	•		•		
selection/consulting	ayamon					•	•	•		
	Enomaly					•	•			
	navica					•	•			
	openlogic					•	•			
	Optaros				•	•	•			
	x-tend					•	•	•		
Other	CiviCRM				•					
	Eclipse				•					
	Mozilla				•					
	OSAF Chandler				•					
	Sourceforge									

Modelos de negocio en empresas ***FLOSS*** (Carlo Daffara, "Business models in FLOSS-based companies http://opensource.mit.edu/papers/OSSEMP07-daffara.pdf)

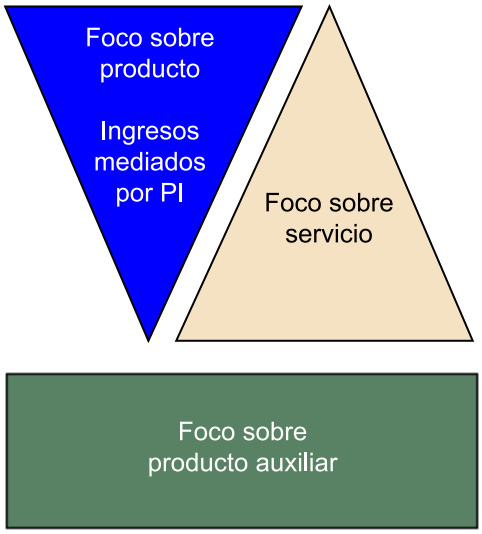
2.4. Propuesta de clasificación

La última clasificación analizada resulta interesante al proporcionarnos datos empíricos sobre empresas reales que están centrando, en la actualidad, su modelo de negocio en torno al software libre. Sin embargo, al igual que Hecker, Daffara propone una caracterización aislada, más que una taxonomía. A continuación propondremos un esquema propio para ordenar e incorporar las ideas que hemos analizado hasta ahora, clasificando los modelos según el grado en el que sus ingresos se derivan de la propiedad intelectual sobre ese software, y según su foco esté centrado en proporcionar productos o servicios:

• Especialistas/Verticales
(Empresas desarrolladoras con un programa libre como producto principal)
– Mixtas OSS/propietarias: Dobles licencias
– Mixtas OSS/propietarias: Núcleo libre, accesorios propietarios
– Puras OSS: "Venta distribuida" de producto libre
– Puras OSS: Servicios sobre el producto
– Software como servicio (SaaS)

• Prestadoras de servicios asociados al software
– Empresas distribuidoras de plataformas
– Grandes integradoras
– Pymes y microempresas de nicho

• Mercados auxiliares
– Hardware
– Otros



Nuestra clasificación, como la de otros autores, se basa en la fuente de ingresos. Sin embargo, además de considerar cómo recuperan la inversión en el desarrollo de software libre distintas empresas, también es importante examinar cómo aprovechan las ventajas un modelo de desarrollo libre puede aportar.

Un modelo de negocio se caracteriza, además de por su fuente de ingresos, por el mercado hacia el que se orienta, cómo elabora y comercializa sus productos, y cómo se relaciona con la competencia. En este sentido, hay un aspecto transversal a cualquier modelo de negocio que cobra especial relevancia gracias al uso del software libre: el concepto de **coopetencia** (***coopetition***).

Coopetencia

Entre otros aspectos diferenciadores frente al software propietario, el empleo de software libre podrá aumentar la calidad de los servicios prestados, contribuyendo a la eliminación de las barreras de entrada, y dibujando un escenario de mayor competitividad y esfuerzo por la diferenciación y la especialización; así como de una competitividad distinta, abierta, cooperativa, en la que las empresas tendrán que cooperar, además de competir, si quieren prosperar. Es-

te concepto empresarial, que en cierta medida está sustituyendo al de "el ganador se lleva todo" en el contexto de una nueva economía de red, se llama coopetencia (*coopetition*).

Coopetencia: Cooperación entre empresas competidoras para buscar escenarios ganador-ganador, bien para incrementar el valor del producto, bien para incrementar el mercado.

Web recomendada

Para más información podéis consultar:

Henry Chesbrough; Wim Vanhaverbeke; Joel West. "Open Innovation: researching a new paradigm"

http://www.openinnovation.net/Book/NewParadigm/Chapters/index.html

En este contexto, las empresas deben examinar cuidadosamente su **ecosistema económico** –clientes, proveedores, competidores y complementadotes– poniendo en marcha estrategias para la creación de nuevas alianzas, y repensando las asociaciones tradicionales.

Este concepto no es exclusivo del software libre, y está extendido a otras áreas. Empresas en una misma industria pueden colaborar entre ellas para ampliar sus mercados, compitiendo más tarde a la hora de segmentarlos.

Ejemplo

Intel invertirá cantidades importantes en ampliar el mercado de los microprocesadores, a pesar de que una parte de esa inversión beneficiará directamente a sus competidores, AMD. En este caso, dada la dominancia de Intel, el porcentaje de su inversión que beneficiará a otros será bastante bajo.

Aunque la coopetencia no sea de aplicación exclusiva en el campo del software libre, sin duda cobra un significado especial en escenarios de desarrollo de código abierto. El hecho de que la competencia se beneficie de la inversión propia es inevitable, por lo que se hace necesario buscar maneras de convertir esta aparente desventaja en una ventaja empresarial. Por otro lado, incorporar a los usuarios (clientes) dentro del proceso de desarrollo, involucrándoles de manera participativa como aliados, es también una característica del modelo de desarrollo del software libre.

El uso de software libre también limitará en gran medida la posibilidad de establecerse como monopolio, además de ofrecer una garantía anti-cautividad. De nuevo, una pregunta clave para cualquier empresa se hace especialmente relevante en un escenario de software libre: ¿cómo crear valor para un cliente extrayendo al mismo tiempo parte de ese valor para la empresa?

3. Modelos de negocio con software libre

En este apartado examinaremos cada modelo de negocio, viendo ejemplos concretos. Es importante tener en cuenta que no se trata de modelos estancos, sino más bien de un continuo difuso. Muchas de las empresas que mencionaremos combinan varios de los modelos, aunque las etiquetemos para sistematizar su estudio.

3.1. Especialistas/Verticales (una aplicación libre como principal producto)

En este apartado incluimos empresas que están produciendo software libre, siendo las promotoras y/o líderes de proyectos concretos. Su implicación con el software libre es, por lo tanto, muy importante, y uno de los aspectos clave en su estrategia empresarial será el manejo de la comunidad, y poder aprovechar las posibilidades de innovación, difusión y trabajo voluntario que ofrece. En esencia, estos modelos mantienen un producto libre para la comunidad, y un producto o servicio relacionado como oferta comercial, y su clave de éxito a menudo estará en conseguir mantener ambas facetas en equilibrio. Según Marten Mickos, consejero delegado de MySQL AB:

> "Las empresas FOSS no funcionarán a no ser que sirvan por igual a aquellos que quieren gastar tiempo para ahorrar dinero, como a aquellos que gasten dinero para ahorrar tiempo".

Este tipo de empresas son las más numerosas en el estudio de Daffara, incluyendo las cuatro primeras categorías (dobles licencias, versiones OSS/propietarias, *badgeware*, y especialistas de producto). Equivaldrían a las empresas de productos vistas en el módulo 3 de esta asignatura, por lo que su problema principal será cómo recuperar la inversión inicial dedicada al desarrollo.

Como hemos visto en las anteriores clasificaciones, una estrategia común es la de obtener ingresos a través de licencias propietarias, que de diferentes maneras se combinan con licencias libres.

También podemos incluir los **modelos que combinan las licencias propietarias cíclicamente**, como los *loss leader* y *sell it*, *free it* de Hecker. **El concepto de *loss leader*** no es exclusivo del software, y es una estrategia de amplia difusión en cualquier sector de actividad: se ofrece un producto de manera gratuita, o a un precio tan bajo que supone pérdidas para la empresa proveedora, como forma de atraer la atención de un gran número de clientes potenciales a los que se les pretende vender otros artículos. En este sentido, tanto los modelos de dobles licencias, como los de producto libre más extensiones propietarias, usan en cierta medida una estrategia de *loss leader*.

Además de fomentar la venta del producto relacionado, en el campo del software una estrategia de código abierto de este tipo traerá varias ventajas, como contribuir a establecer su tecnología como estándar *de facto*, atraer mejoras y complementos que hagan al producto más atractivo, generar simpatía entre una audiencia que puede incluir a los clientes potenciales del producto relacionado, y reducir los costes de mantenimiento del proyecto.

A continuación trataremos en detalle el modelo de dobles licencias y el de producto principal libre con accesorios propietarios. Omitimos otros modelos en los que el producto principal no es libre, porque en realidad representan modelos de negocio basados en software propietario: la liberación de código sólo supone una estrategia de negocio complementaria para mejorar la posición de su producto principal propietario.

Daffara también nos muestra un número importante de empresas que lideran proyectos de desarrollo con licencias puramente libres, y que obtienen sus ingresos del **ITCS** (*installation/training/support/consulting*). Quizás este grupo es uno de los que más modelos distintos puede englobar, estando sus ingresos enmarcados en una categoría bastante vaga. En este sentido, será importante examinar con más cuidado a qué mercados se dirigen y qué diferenciación presentan ante otros productos equivalentes, aparte de la de mejor conocimiento.

3.1.1. Modelos mixtos: licencias dobles

Este modelo se basa en la distribución de un producto bajo dos licencias distintas: una licencia propietaria tradicional, y una licencia libre restrictiva (tipo GPL). De esta manera, si alguien quiere generar un trabajo derivado, y redistribuirlo sin el código, puede hacerlo, pero deberá pagar una licencia. De lo contrario, todos los trabajos derivados deben redistribuirse con el código.

Michael Olsen, gerente de Sleepycat Software Inc., productores de BerkeleyDB, describe su modelo de doble licencia de la siguiente manera:

> "La licencia *open source* de Sleepycat permite el uso de Berkley DB [...] sin coste, bajo la condición de que si se usa el software en una aplicación que más tarde se redistribuya, el código completo de la aplicación debe estar disponible, y debe poder ser redistribuido de nuevo libremente bajo condiciones razonables. Si no se quiere ofrecer el código fuente de una aplicación derivada, se puede comprar una licencia de Sleepycat Software."
>
> S. Comino; F. M. Manetti. "Dual licensing in open source markets". Disponible en: http://opensource.mit.edu/papers/dual_lic.pdf

Esta estrategia resulta apropiada cuando una parte relevante de la demanda se genera por usuarios comerciales que necesitan el software para embeberlo en sus propios productos. Estos clientes usan el producto comprado como un *input* para producir otro software, bien como un producto final, o como parte de una tecnología más compleja producida y vendida por este cliente comercial.

Sea para poder vender sus productos derivados bajo un esquema tradicional propietario, o porque el software que genera es parte fundamental de su diferenciación, este cliente necesitará poder cerrar el código que genere, y pagará por ello.

Estos modelos segmentan a sus usuarios en dos grupos: la comunidad –todos aquellos usuarios que estén satisfechos con licencias libres, y usen el producto bajo esos términos– y clientes corporativos sensibles a los términos de reciprocidad de licencias libres.

Sin embargo, el mantenimiento de una comunidad de personas colaboradoras en torno al producto puede resultar problemático. Por un lado, la obtención de ingresos directos por el producto puede influir en la motivación de voluntarios que contribuyen sin recibir nada a cambio. Por otra parte, las compañías que lo implementan deben recoger de manera formal de sus voluntarios la asignación de *copyrights*, para evitar problemas en el futuro de colaboradores descontentos reclamando su parte de ingresos en concepto de licencias por el producto que contribuyeron a desarrollar.

En la práctica, las empresas que basan su modelo en la doble licencia no se benefician extensamente de las ventajas de conseguir aportaciones externas al desarrollo, consiguiendo sólo resolución de errores a pequeña escala y algún parche por parte de la comunidad. El equipo principal de desarrollo típicamente estará dominado, casi al 100%, por empleados de la compañía.

Otro problema que pueden encontrar estos modelos es que sus clientes puedan construir sus extensiones propietarias, sin necesidad de modificar el código original, por lo que podrán usar la versión con la licencia libre, y mantener sus adiciones como una aplicación separada e independiente.

A menudo, estas empresas combinan los ingresos generados por las dobles licencias con otras actividades, como la prestación de servicios que veremos en apartados posteriores. Ejemplos de este modelo son Funambol, MySQL, Sleepycat DB, y TrollTech/NOKIA.

El caso de Funambol

Nombre de la compañía	Funambol, Inc.
Sede	Redwood City (Estados Unidos)
Página web	www.funambol.com
Fecha de creación	2001
Nº de personas empleadas en 2007	40
Volumen de ventas en 2007 (millones)	$4,8

Datos corporativos de Funambol, Inc. Tabla elaborada a partir de estadísticas de Hoovers (http://www.hoovers.com)

Web recomendada

Para más información podéis ver:
http://www.funambol.com/blog/capo/2006/07/my-honest-dual-licensing.html

Funambol es una corporación norteamericana dedicada, según su lema, al "mobile 2.0 messaging powered by open source". La empresa desarrolla un servidor de aplicaciones móviles (proporciona *push e-mail*, libro de direcciones y calendario, sincronización de datos, y servidor de aplicaciones para dispositivos móviles y PC), además de una plataforma de desarrollo para aplicaciones móviles, ambas desarrolladas bajo el nombre de "Funambol".

Comercializa su base de código bajo dos licencias: la AGPLv3 para su "Community Edition", y una licencia propietaria comercial para su "Carrier Edition", pero combina esta estrategia con la de proporcionar funcionalidades adicionales necesaria para grandes implementaciones en la versión cerrada, además de servicios basados en la "Carrier Edition".

En este caso, Funambol ha elegido como licencia la "Affero" GPL, que le proporciona una protección extra frente a usos comerciales de sus aplicaciones en la modalidad de software como servicio (SaaS). Como vimos en el módulo tres de esta asignatura, la GPL permite la modificación del código sin su redistribución, siempre que la aplicación en sí tampoco sea redistribuida, como ocurre con la prestación del software como servicio. La "Affero" GPL resuelve el problema de este vacío, requiriendo la redistribución del código fuente también cuando la funcionalidad del software se ofrezca mediante el modelo "SaaS".

La naturaleza del software lo hace idóneo para el modelo de doble licencia, ya que previsiblemente resultará atractivo para otras empresas que quieran desarrollar aplicaciones cerradas sobre su plataforma, como operadores de telefonía móvil, fabricantes de dispositivos y otras compañías de software. Gracias al uso de la AGPL, aquellas empresas que usen Funambol como base de sus ofertas "SaaS" también deberán pagar si no quieren redistribuir el código. Entre sus clientes se encuentran Vodafone, Earthlink y Computer Associates.

Funambol intenta explotar al máximo las necesidades de grandes clientes corporativos con la incorporación de funcionalidad adicional en su versión comercial y la prestación de servicios. Para evitar los problemas derivados del modelo "núcleo libre + accesorios propietarios" que veremos más adelante, se asegura de que la funcionalidad cerrada sólo sea interesante en escenarios de grandes implementaciones corporativas, por lo que su comunidad de usuarios libres no sentirán la necesidad de desarrollar esa funcionalidad por su cuenta.

Fabrizio Capobianco, gerente de Funambol, justifica el modelo de doble licencia en "My Honest Dual Licensing" como el modelo más "honesto" de mantener los principios del desarrollo de software libre, compatibilizándolo con la necesidad empresarial de generar un beneficio.

Sin embargo, como hemos visto anteriormente, definir una fuente de ingresos viable no garantiza el éxito de ninguna empresa, y el uso de software libre permitirá poner en marcha estrategias cualitativamente distintas a las de un modelo basado en software propietario. Funambol es un caso muy ilustrativo en este sentido, en el que la empresa tuvo que redefinir tanto sus prácticas de marketing como la población objetivo de éstas, antes de tener un modelo de negocio viable.

En sus inicios, Funambol intentó construir un modelo clásico de vendedor de software en torno a su producto de software libre. La compañía había desarrollado Sync4j, que permitía a desarrolladores construir aplicaciones para dispositivos móviles bajo el paradigma "a veces conectado" (la aplicación puede funcionar desconectado, sincronizando los datos cuando recupera conexión). Identificó como sus clientes potenciales grandes compañías y operadores inalámbricos, que dado el gran número de trabajadores con los que cuentan, y aprovechando las crecientes posibilidades de movilidad, necesitarían sincronizar datos entre diversos dispositivos móviles y sus servidores corporativos.

Para hacer llegar su producto a estos clientes, Funambol decidió seguir una estrategia de ventas proactiva, poniendo un gran esfuerzo en su equipo de marketing y ventas, que trataba de acceder a ellos de forma directa, mediante campañas telefónicas.

Su éxito fue muy limitado. Funambol no consiguió cumplir las expectativas de ventas que tenía, encontrando que las grandes corporaciones eran reacias a tratar con una pequeña nueva empresa como ella. Por otro lado, los ciclos de venta se hacían muy largos, y pronto se hizo patente que para mantener esta estrategia se requería un equipo de ventas y marketing mucho más grande del que Funambol se podía permitir.

Funambol identificó pronto que sus problemas partían de esta estrategia de ventas activa, tradicional en el mundo del software propietario, pero que supone una barrera de entrada que pocos consiguen cruzar: para acceder a un conjunto de clientes potenciales compuesto por grandes corporaciones, a menudo hace falta tener una gran capacidad

de marketing y ventas, además de contar con un tamaño y reputación suficientes para transmitir la necesaria confianza.

El uso de software libre permitía revertir esta estrategia, centrándose en un marketing reactivo, en respuesta a la iniciativa del cliente. En este nuevo escenario, serían los clientes potenciales los que buscarían a Funambol, dejando a la compañía el papel de estar atenta para identificarles una vez éstos se ponían en contacto con ella.

La efectividad de esta estrategia sólo dependía de un factor: el número de descargas de su producto. Con un número de descargas suficiente, identificaron el siguiente ciclo de venta típico (mucho más corto que el que experimentaron con su anterior estrategia):

1) El usuario potencial accede a la web de Sync4j para buscar información sobre el producto y documentación técnica.

2) El usuario descarga el producto.

3) Más tarde, se suscribe a la lista de correo, buscando más información.

4) Tras un uso intensivo de su producto (normalmente en proyectos I+D), el cliente contacta con Funambol para preguntar por precios y condiciones de licencia. Internamente, se le clasifica como cliente potencial.

5) Finalmente, pide un presupuesto y oferta oficial, y puede convertirse en un Cliente Funambol.

(Fabrizio Capobianco; Alberto Onetti. "Open Source and Business Model Innovation. The Funambol case". Disponible en: http://oss2005.case.unibz.it/Papers/4.pdf)

El factor clave para seguir alimentando este ciclo es, como hemos dicho anteriormente, mantener el número de descargas del producto alto. El ciclo se retroalimenta, generando por sí mismo más descargas, por lo que tras un esfuerzo inicial, este mecanismo tomaría suficiente inercia para funcionar casi por sí solo.

Para lograrlo, Funambol se centró en **crear comunidad en torno a su producto**, focalizando sus esfuerzos de marketing sobre los usuarios de su versión libre, tanto los expertos como aquellos con menos habilidades técnicas. Aunque esta estrategia no se orienta hacia sus clientes generadores de ingresos de forma directa, resultó ser mucho más barata y eficiente.

La empresa se dedicó a dar a conocer el producto entre desarrolladores, participando en foros de desarrollo, en listas de correo, revistas especializadas, conferencias, creando alianzas con organizaciones sin ánimo de lucro de promoción del software libre, o creando sinergias con otros productos de código abierto bien establecidos. De cara a los usuarios más inexpertos, fue necesario asegurarse de que el producto fuera de fácil instalación, con suficiente documentación accesible desde su página web. La empresa vio cómo, al trabajar estos dos últimos factores, el número de descargas del producto se incrementaba notablemente, poniendo en marcha, de esta manera, su ciclo generador de ventas.

3.1.2. Modelos mixtos: núcleo del producto libre y accesorios propietarios

En este modelo ("Split OSS/commercial releases", según Daffara), existen dos versiones distintas de un programa, una versión básica libre, y una versión comercial propietaria, basada en la anterior, pero con funcionalidad adicional implementada a través de *plugins* o accesorios. La versión libre debe emplear una licencia de tipo MPL o BSD, que permita la combinación para crear un producto cerrado.

El problema principal de este modelo será mantener el producto libre suficientemente interesante, sin restarle valor al producto propietario generador de ingresos. También corre el riesgo de que la comunidad en torno al producto decida por su cuenta desarrollar la funcionalidad de la versión propietaria, haciendo difícil la generación de ingresos por su venta.

En este tipo de modelos, se distinguen dos clases de usuarios: aquellos que estarían dispuestos a pagar a cambio de obtener un producto con alguna funcionalidad adicional (medianas y grandes empresas), y aquellos muy sensibles al precio, como pequeñas empresas, microempresas o usuarios individuales. Al combinar versiones libres y propietarias, se consigue una mayor adopción de la solución propuesta, sin perder por ello la captura de ingresos a través de las versiones propietarias. Como hemos visto en módulos anteriores, en un escenario de "the winner takes it all" común en el mundo del software, la estrategia basada en una amplia adopción resulta de gran importancia.

En este sentido, parte de los mismos principios de segmentación de sus usuarios que el modelo de dobles licencias, pero corre mayor riesgo de perder la simpatía de la comunidad, que no tiene acceso a todo el código fuente.

Un ejemplo de este modelo es **Sendmail Inc.**, que comercializa una constelación de productos propietarios en torno al servidor libre sendmail. Otros ejemplos son Hyperic (IT Operations/Monitoring), SourceFire (SNORT comercial), Zimbra/Yahoo (mensajería, *groupware*), y XenSource/Citrix (virtualización).

El Caso de Sendmail

Nombre de la compañía	Sendmail, Inc.
Sede	Emeryville, CA. (Estados Unidos)
Página web	www.sendmail.com
Fecha de creación	1997
Nº de personas empleadas en 2007	125
Volumen de ventas en 2007 (millones)	$23

Datos corporativos de Sendmail, Inc. Elaborada a partir de estadísticas de Hoovers (http://www.hoovers.com)

Al estudiar modelos de negocio basados en software libre, a menudo pensamos en corporaciones que deciden abrir el código como ventaja competitiva para ampliar su cuota de mercado. Sendmail resulta un caso interesante, en el que el proceso tiene lugar de forma inversa: con orígenes libres y sin ánimo de lucro, el establecimiento de una iniciativa comercial en torno al proyecto no sólo pretende obtener ingresos del desarrollo, sino mantener la posición dominante del proyecto en su sector, y ampliar su base de usuarios.

Sendmail es un agente de transferencia de correo ("mail transfer agent", MTA) y supone uno de los ejemplos más conocidos de proyectos nacidos en el seno de comunidades de software libre. En 1998, se estimaba que el 80% de todo el tráfico de correo electrónico se enviaba a través de Sendmail. Actualmente, continúa siendo el MTA más popular en Internet, aunque ha perdido parte de sus usuarios en Microsoft Exchange Server, Exim y

Postfix. Igualmente destacable es el largo periodo de vida que está teniendo el producto, y que tiene su origen en desarrollos iniciados en la década de los setenta.

Eric Allman desarrolló la primera versión de Sendmail a principios de la década de los ochenta, en el seno de la Universidad de Berkeley, a partir de trabajo previo en el programa Delivermail, y en 1997 fundó Sendmail, Inc. La estrategia de la empresa se centraba en vender funcionalidades adicionales relacionadas con Sendmail de manera propietaria (por ejemplo, interfaces más amigables), además de en proporcionar servicios complementarios. Al mismo tiempo, la compañía se esforzó por mantener la continuidad del desarrollo de Sendmail de forma abierta, proporcionando servicios de alojamiento y recursos humanos para el desarrollo.

Al crear la compañía, Allman esperaba no sólo desarrollar una actividad empresarial, sino proteger la posición dominante de Sendmail, que estaba en peligro por la aparición de formatos propietarios que amenazaban el estándar abierto SMTP. La empresa orientó sus esfuerzos hacia el entorno corporativo, ofreciendo no sólo servicios de integración y soporte, sino un producto que se ajustara más a sus necesidades. Las extensiones creadas por la compañía se centran en ofrecer interfaces gráficas y mayor facilidad de gestión, y se comercializan de manera propietaria.

"Sendmail, Inc. develops commercial products and services for ISPs and enterprises for whom email is mission critical, while continuing to drive innovation and standards through Open Source software development."

Sendmail, Inc

Podemos considerar que la creación de Sendmail, Inc. fue el paso necesario para cruzar el "abismo" (*the chasm*), y lograr la adopción del producto por las mayorías pragmáticas y conservadoras. Sin embargo, para Allman era importante mantener la funcionalidad original de Sendmail libre, para lo que se creó Sendmail Consortium, entidad sin ánimo de lucro responsable del desarrollo de la versión libre. De esta manera, capitaliza las ventajas de un modelo de desarrollo libre, tanto en lo que a contribuciones y reducción de gastos se refiere, como a innovación y evolución del producto.

Allman aprovechó así "el abismo" para poder vender de forma propietaria extensiones a su producto, sin correr el peligro de la ramificación de su proyecto. Siguiendo el modelo de Moore, la comunidad en torno al proyecto libre Sendmail estaría formada por los innovadores y los entusiastas tecnológicos, interesados en la funcionalidad en bruto, y en las propuestas nuevas. Los clientes comerciales, sin embargo, serían los pragmáticos y conservadores, con necesidades y objetivos muy distintos. Las extensiones propietarias, centrándose en las funcionalidades de empaque y acabado del producto (facilidad de uso, interfaces gráficas, estabilidad, etc.), no sólo no resultan interesantes para los innovadores, sino que les pueden llegar a parecer innecesarias. La presencia de ese abismo entre los intereses de la comunidad y los clientes comerciales permite la co-existencia de la versión central libre, y la extendida propietaria, sin peligro de bifurcaciones (*forks*), ya que la comunidad no tendrá interés en las extensiones del otro lado del abismo.

3.1.3. Modelos libres: "Venta distribuida" del producto

Es habitual asumir que al licenciar de una manera libre un producto se pierde la oportunidad de obtener ingresos directos vinculados a la propiedad intelectual sobre él, haciéndose necesario explotar otros productos o servicios complementarios.

Web recomendada

http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/673/583

Sin embargo, elegir una licencia libre para un trabajo no tiene por qué significar la renuncia a la obtención de ingresos relacionados de forma directa con ese producto. La idea de que nadie pagará por algo que eventualmente puede conseguir de forma gratuita, aunque extendida, no es fiel a la realidad. Mucha gente está dispuesta a pagar una cantidad pequeña por un trabajo que valora si cree que ese dinero irá a las autoras originales. Si un proyecto es suficientemente exitoso, puede recibir pequeñas aportaciones de mucha gente, llegando quizás a financiar su creación, de la misma manera que un artista callejero, sin cobrar entrada, puede recaudar lo suficiente como para hacer rentable su in-

versión de tiempo y esfuerzo. Esta es la idea que subyace en el *The Street Performer Protocol and Digital Copyrights*, de John Kelsey y Bruce Schneier al proponer un mecanismo de financiación distribuida de trabajos digitales, en el que la autor no realizaría su obra hasta haber recaudado financiación suficiente.

Distintos mecanismos para articular esta financiación directa y distribuida se han descrito y puesto en marcha en el contexto del desarrollo de software, desde donaciones y recompensas (*bounties*), a la creación de mercados vía web, para poner en contacto a desarrolladores y clientes potenciales, siguiendo un esquema de bonos similar al descrito por Chris Rasch en *The Wall Street Performer Protocol*.

Web recomendada

http://www.firstmonday.org/issues/issue6_6/rasch/index.html

Las donaciones son el mecanismo más sencillo de este tipo de financiación, pero demasiado inciertas para los creadores, que necesitan la seguridad de un ingreso antes de invertir su tiempo. En los sistemas de bonos y recompensas, las personas interesadas en una funcionalidad concreta ofrecen una recompensa por que sea implementada. Una vez la recompensa total (a la que pueden contribuir diversas personas) es suficiente para un desarrollador, puede ofrecerse a realizarla, cobrando una vez se haya completado. Algunos de estos sistemas se basan en la confianza entre el equipo de desarrollo y los usuarios, sin exigir garantías de pago, otros proponen el establecimiento de algún tipo de intermediario neutral.

El servidor Cherokee

Este servidor decidió implementar un sistema de recompensas con el objetivo principal de atraer a nuevos desarrolladores al proyecto. Además de recompensar el esfuerzo, el ofrecer un retorno económico atraería a más personas a la comunidad de desarrollo, haciendo que el proyecto creciera.

Mercados virtuales

Varios intentos de creación de "mercados virtuales" de software basados en este tipo de financiación se han puesto en marcha. Algunos de los que están en funcionamiento en la actualidad son BountyCounty (http://bountycounty.org/), MicroPledge (http://micropledge.com/) y BountySource (https://www.bountysource.com/).

La clave del éxito en estos escenarios puede encontrarse más en las facilidades de pago ofrecidas que en la disposición de pagar de los usuarios:

> "La mayoría de las personas estarán dispuestas a pagar una cantidad pequeña sobre una mayor, si ya han sacado su cartera, y si piensan que es por una buena razón. Cuando las personas dejan de hacer pequeñas donaciones voluntarias a una causa que les gusta, suele ser más por las molestias (escribir un cheque, mandarlo por correo, etc.), que por el dinero en sí."
>
> (Karl Fogel. "The Promise of a Post-Copyright World". Disponible en: http://www.questioncopyright.org/promise)

Aunque muchos proyectos ponen en práctica estas ideas para obtener financiación complementaria, es difícil encontrar escenarios corporativos en los que el grueso de sus ingresos se obtenga a partir de estos mecanismos.

Por un lado, en el contexto del software este tipo de financiación se puede hacer más difícil al no existir una identificación y simpatía con las personas autoras tan fuerte como la que se deriva de otros trabajos creativos.

Por otro lado, este modelo seguramente tendrá más éxito si se trata de un proyecto de software libre sin ánimo de lucro e integrado en su totalidad por voluntarios, que despertará más fácilmente las simpatías de sus usuarios. Una empresa que quiera utilizarlo con éxito, seguramente tendrá que buscar un

reconocimiento previo, mediante transparencia y confianza, mostrando que su ánimo de lucro no es a costa de todo y revertirá en el bien común (más adelante veremos modelos de negocio basados en estos principios).

Estos esquemas presentan un modelo económico más directo, se eliminan intermediarios, y se proporciona mayor cercanía entre usuarios y desarrolladores. Desde cierto punto de vista, podrían considerarse como la manera natural de financiación de un proyecto de software libre: de la misma manera que voluntarias contribuyen en distinta medida y en distintos aspectos del ciclo de desarrollo del software, los usuarios pueden formar parte del propio proyecto contribuyendo con un aporte económico de acuerdo a sus posibilidades e intereses.

3.1.4. Producto libre más servicios asociados

Las empresas de esta categoría implementan una estrategia del tipo "mejor conocimiento" y "mejor código", desarrollando un producto libre, y ofreciendo servicios sobre él como motor de ingresos.

En este apartado incluimos tanto los especialistas de producto como los *badgeware* del estudio de Daffara, ya que ambos representan el mismo modelo de negocio. Por otro lado, aunque las licencias de tipo *badgeware* incluyen una restricción adicional de atribución, mantienen las características esenciales de apertura y libertad de conocimiento, pudiendo generar las mismas ventajas a través de sus comunidades de desarrollo que aquellos que empleen licencias sin esta restricción. Probablemente aquellos ejemplos enmarcados en *badgeware* deseen poner en marcha también cierta estrategia de marca, por lo que depositan una importancia especial en la atribución a la hora de redistribuir los productos que generan.

Este modelo tiene varios problemas, como pocas barreras de entrada en el negocio –cualquier empresa puede adquirir conocimiento sobre el producto y ofrecer servicios– o los problemas a la hora de conseguir contratos de soporte –las empresas clientes pueden preferir continuar con sus empresas de servicios o consultoría habituales, o contratar empresas proveedoras que ofrezcan soporte sobre toda su infraestructura de nuevas tecnologías, y no sólo frente a un producto concreto.

Otro problema al que se suelen enfrentar estos modelos para generar ingresos a partir de servicios es el de los innovadores y entusiastas: cuando un nuevo producto entra en el mercado, los primeros usuarios suelen ser personas con competencias técnicas, que no contratarán servicios de soporte sobre él, y preferirán adquirir el conocimiento necesario por sí mismas. Este modelo, por lo tanto, necesitará ofrecer un producto extendido y que transmita fiabilidad, para alcanzar un mercado potencial capaz de pagar por obtener servicios sobre el producto.

El éxito de este tipo de modelos de negocio es cuestionado por algunos autores (por ejemplo, Perens), sin embargo, también hay un número importante de empresas que se asientan sobre este modelo, habiendo atraído cuantías importantes de capital riesgo. Para llevar a cabo un modelo de negocio sostenible, sin embargo, tendrían que acometer los problemas mencionados anteriormente.

Entre los modelos de especialistas verticales prestadores de servicios, encontramos a Alfresco (gestión de contenidos), Compiere (ERP, CRM), vTiger, y Openbravo.

El caso de Openbravo

Nombre de la compañía	Openbravo, S. L.
Sede	Pamplona (España)
Página web	www.openbravo.com
Fecha de creación	2001
Nº de personas empleadas en 2007	26 a 50
Volumen de negocio en 2007	Hasta 300.000 €

Datos corporativos de Openbravo. (Obtenidos de http://www.camerdata.es)

Openbravo supone un ejemplo interesante de este tipo de modelo. La empresa, fundada en 2001, se centra en el desarrollo de dos aplicaciones libres para PYMES- OpenbravoERP (planificación de recursos empresariales), y OpenbravoPOS (punto de venta), que tratan de satisfacer las necesidades de gestión y planificación, y de terminal de punto de venta de pequeñas y medianas empresas, respectivamente. El código fue publicado en el año 2006, y actualmente está entre los proyectos más activos en el ranking de Sourceforge.

La empresa ha atraído cantidades importantes de capital riesgo, contando como inversores con Amadeus, Gimv, Adara, y SODENA (Sociedad de Desarrollo de Navarra), que ha invertido 5 millones de euros en la compañía.

Su estrategia empresarial está orientada a convertirse en producto líder en el sector, intentando que OpenbravoERP sea el software gestión de referencia entre las pymes. Para lograrlo, la empresa está explotando al máximo las posibilidades que le brinda el software libre, mediante un cuidado manejo de la comunidad, y la aplicación del concepto de coopetencia.

De una manera similar a la que hemos visto en el caso de Funambol, Openbravo vio que por sí misma no tenía capacidad para difundir y distribuir su producto entre sus usuarios potenciales. Aunque OpenbravoERP y OpenbravoPOS no se dirigen a grandes corporaciones, sino a pymes, para lograr sus objetivos estratégicos de convertirse en líder en el sector, el producto debía llegar a innumerables pequeñas y medianas empresas en todo el mundo.

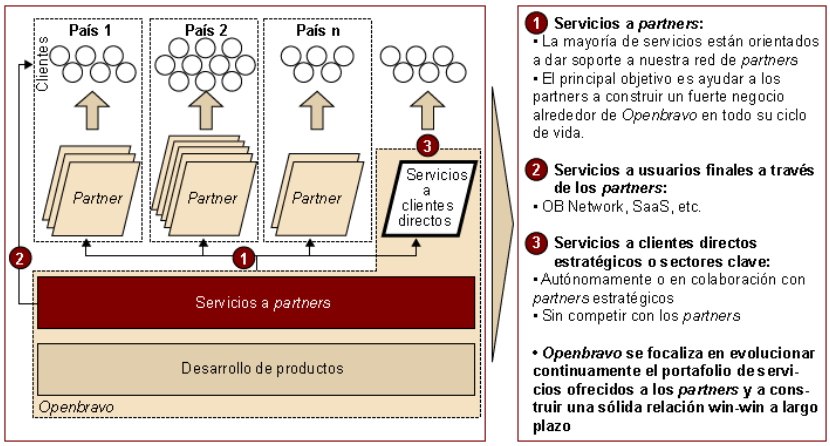
Además del tamaño necesario para llevar a cabo una campaña de marketing de esta escala, la empresa también era consciente de las dificultades que podía encontrarse a la hora de competir por prestar servicios directamente a sus usuarios finales, que podrían preferir a empresas locales, o que ofrecieran soluciones integrales, y no sólo frente a un producto.

Para solucionar estas barreras, Openbravo posicionó a estas empresas no como competidoras, sino como colaboradoras. De esta manera, admite que no porque sean los desarrolladores del producto, hayan de ser los idóneos a la hora de prestar servicios sobre él a los usuarios finales. Su misión sería la de crear un buen producto que pudiera ampliar mercados, generando nuevas oportunidades de ingresos para empresas de servicios informáticos, que podrían completar su oferta con OpenbravoERP y OpenbravoPOS.

De esta manera, Openbravo define como motor de ingresos la prestación de servicios a otras empresas de servicios informáticos, intermediarias entre ella y los usuarios finales.

Estas empresas constituirán una red de *partners*, que llevarán a cabo las tareas de implementación de OpenbravoERP y OpenbravoPOS en las pymes.

Openbravo ofrece a sus *partners* diversos servicios (soporte, formación), poniendo en marcha un esquema de consultoría piramidal similar al descrito en el módulo 3 de esta asignatura, además de transferirles fiabilidad y confianza. Al estar respaldadas por los desarrolladores del producto, pueden explotar la estrategia de "mejor conocimiento", y "mejor código" en sus mercados.



Openbravo: Oportunidad de negocio y vías de crecimiento (obtenida de la presentación de Openbravo en WhyFLOSS Madrid 2008 "Openbravo: keys to success in free software application development". http://www.whyfloss.com/es/conference/madrid08/getpdf/49).

Openbravo establece así una estrategia de *coopetencia*, dejando a empresas de servicios la posibilidad de explotar Openbravo en el contexto de sus mercados naturales, pero beneficiándose de una mayor difusión de su producto, y obteniendo ingresos directamente de sus *partners*. Hasta el momento, ha tenido bastante éxito con esta estrategia, contando en la actualidad con 85 *partners* en todo el mundo.

3.1.5. Software como servicio

Las empresas desarrolladoras de un producto también podrán explotarlo mediante el paradigma de software como servicio. En lugar de ofrecer servicios de instalación y soporte, la empresa se hace cargo de toda la infraestructura de hardware y software, ofreciendo directamente la funcionalidad a través de Internet. Los ingresos generados, de naturaleza recurrente, tomarían la forma de subscripciones al servicio.

Collabnet: software como servicio

Un buen ejemplo de este tipo de modelo lo presenta **Collabnet**. Ofrece servicios para el desarrollo colaborativo de software (control de versiones, seguimiento de incidencias, comunicación, etc.), que genera gracias a, entre otros, la plataforma de control de versiones Subversion. En este caso, además de mantener el código abierto, la empresa dedica esfuerzos importantes al mantenimiento de la comunidad, de manera que su trabajo sobre el proyecto no es más que una aportación (aunque sea grande) dentro de una comunidad libre. Otros ejemplos de empresas que comercializan sus productos siguiendo el modelo de "software como servicio" son sugarCRM, SocialText y JasperSoft.

Estas empresas, en la modalidad de "software como servicio", no encontrarán dificultades adicionales respecto a sus equivalentes propietarias en la generación de ingresos, ya que en este caso las ventas no se derivan de derechos de *copyright* sobre el producto. El que un cliente pueda descargar, instalar, configurar, alojar y mantener la aplicación supondrá más una herramienta de mar-

keting y difusión que una pérdida de ingresos. Como hemos visto anteriormente, el cliente corporativo estará dispuesto a pagar una cantidad a cambio de ver sus problemas resueltos.

Sin embargo, liberar todo el código presentará problemas en cuanto a diferenciación, y posibilidad de entrada a competidoras. Cualquier empresa con capacidad técnica e infraestructura suficiente podrá ofrecer un servicio similar, a partir de la disponibilidad del código. Ante este problema, la empresa que ha desarrollado el producto podrá basar su diferenciación en el "mejor conocimiento", y "mejor código", además de poder ganar las simpatías de la comunidad. Por otra parte, ante empresas competidoras que decidan contribuir también al desarrollo, se podrán poner en marcha mecanismos de *coopetencia,* colaborando en la ampliación del mercado, y segmentándolo más tarde mediante la especialización.

De forma análoga a las estrategias mixtas OSS/propietarias que hemos visto anteriormente, algunas empresas de esta categoría pondrán en marcha soluciones que incorporen alguna limitación sobre su código, principalmente manteniendo una pequeña parte del código cerrada, sobre la que basarán su diferenciación.

3.2. Servicios asociados al software libre

Considerando los servicios asociados al software libre, son posibles multitud de negocios ya que, en general, cualquier modelo de servicios basado en software propietario (como los que vimos en el módulo 3) será extrapolable de una forma bastante directa al software libre. Todos los pasos descritos en la cadena de creación e implementación de una solución tecnológica seguirán siendo viables en un contexto de aplicaciones abiertas. Sin embargo, el empleo de software libre amplía las posibilidades y los factores de diferenciación de modelos de negocio centrados en servicios.

Como principio básico diferenciador, podemos destacar la **ausencia de costes derivados de licencias**, que proporciona una clara ventaja competitiva frente a soluciones propietarias. Aun así, para aprovechar este factor, será importante que la solución propuesta sea más barata a largo plazo (considerar el "*total cost of ownership*"), y ofrezca un grado de calidad al menos equivalente a competidores propietarios. También es importante destacar que una empresa de servicios en torno al software libre debería resultar más atractiva a sus clientes por la reducción de la posibilidad de situaciones de captura (*lock-in*): estos proveedores no podrán contar con la continuidad de los ingresos en una situación con clientes cautivos, sino que deberán basarse en la **continua prestación de servicios de calidad**.

Por otro lado, que un software sea libre no significa que sea accesible a todo el mundo. El mercado para empresas de servicios no disminuirá por la disponibilidad de aplicaciones libres o eventualmente gratuitas, ya que el trabajo

de selección, instalación, formación y soporte siempre será necesario en entornos corporativos, y resultará más interesante si el presupuesto de licencias se dedica a un mejor servicio.

Como regla general, este tipo de empresas participan en varios proyectos, aunque no de manera intensiva en ninguno de ellos. Algunas contribuirán, como en el caso de los distribuidores de plataformas, en la solución de errores, especialmente en áreas de interés de sus clientes, así como en tareas de integración y de conseguir la compatibilidad entre distintas aplicaciones. Otras, como las centradas en consultoría y selección (y sin capacidad de desarrollo), no contribuirán a los proyectos sobre los que se sustentan, ya que normalmente su trabajo se mantiene privado y no será visible al público. En estos casos, sin embargo, se puede producir un retorno en forma de promoción y adopción de la solución sobre la que trabajen.

El rango de modelos posibles en esta categoría es enorme (diferenciación en cuanto a tamaño, a segmentación de solución -horizontal o vertical- a segmentación del sector, especialización en algún servicio en concreto: desarrollos a medida, selección, consultoría, integración, formación, etc.), y la mayoría de las empresas ofrecerán una combinación de los servicios posibles. Vamos a examinar primero las características especiales del software libre en las distintas fases de implementación de una solución tecnológica, y más adelante nos centraremos en tipologías concretas de modelos de negocio, que agrupan ciertos servicios de una manera particular.

Desarrollos a medida

El software libre ofrecerá a las empresas un término medio ante la cuestión de "**comprar o desarrollar**". Estas empresas podrán partir de un producto estándar libre, y ya sea de forma interna, o a través de una empresa de desarrollo, construir las adaptaciones necesarias a sus necesidades. Tanto las empresas de servicios que examinaremos a continuación, como las anteriores orientadas a productos, recibirán ofertas para llevar a cabo este tipo de personalizaciones. Sin embargo, realizar estas adaptaciones de forma privada, y sin intentar que sean incorporadas al proyecto principal puede resultar problemático a la hora de mantener las adaptaciones compatibles con las siguientes versiones. En este sentido, trabajar junto con la comunidad, diseñando las nuevas funcionalidades de manera que puedan ser interesantes para más personas, e incorporarlas al código principal del proyecto ahorrará mucho trabajo y complicaciones.

Selección

La presencia de infinidad de aplicaciones, al alcance (económico) de cualquier empresa, hace de la selección una tarea fundamental. No sólo será necesario encontrar productos que cumplan de la forma más ajustada las necesidades de la empresa cliente, sino también evaluar la salud de determinados proyectos, el ritmo de solución de errores y nuevos lanzamientos, y su estabilidad.

Para entornos corporativos, un proyecto con mucho movimiento, y un rápido ritmo de incorporación de mejoras puede no ser el mejor, sino que eventualmente puede resultar más adecuado un producto estable y que no vaya a cambiar sustancialmente con el tiempo.

Instalación e integración

Lectura complementaria

D. Woods; G. Guliani (2005). *Open Source for the Enterprise: Managing Risks, Reaping Rewards*.

Aunque esta fase también genera necesidades en entornos comerciales, el software libre presenta una oportunidad de negocio especial en este campo: su falta de empaquetamiento y acabado final. En *Open Source for the Enterprise*, Woods y Guliani aluden al concepto de "productización" como una de las carencias principales del software libre de cara a una extensa adopción. Con el término se refieren al grado en el que la aplicación ha sido empaquetada y preparada para usuarios finales, con el desarrollo de instaladores automáticos, de interfaces gráficas de configuración y una documentación suficientemente extensa que, en suma, permita su instalación y manejo por parte de usuarios inexpertos.

Como regla general, el software comercial está más empaquetado y acabado que el software libre desarrollado de manera voluntaria. Los *scripts* de instalación, las interfaces administrativas y la documentación suelen ser más completos para un producto propietario comercial, que para un producto de software libre de la misma edad. Mientras que para los entusiastas tecnológicos esta falta de terminación del producto no es importante, o incluso es atractiva (permite realizar el ajuste y la administración de una manera más directa y personal), para cruzar el abismo y conseguir al cliente corporativo, el software libre debe alcanzar un mayor grado de empaquetamiento y acabado. Según Woods y Guliani:

> "Una amplia simplificación sobre el software abierto frente al comercial es que el software de código abierto representa principalmente una inversión de tiempo, y el software comercial representa principalmente una inversión de dinero. Cualquier organización que pretenda usar software libre deberá reservar tiempo para dedicar a la investigación y la experimentación."
>
> Dan Wods y Gautam Guliani. "Open source for the enterprise"

Esta inversión de tiempo en completar una aplicación o selección de aplicaciones de código abierto presenta una importante oportunidad de negocio tanto para los integradores como para los desarrolladores de plataformas. En este sentido, una buena simbiosis se podría establecer entre el sector privado y los proyectos de software libre sin ánimo de lucro, en la que la inversión se dedicaría a realizar trabajos más monótonos, dejando a la comunidad voluntaria el trabajo más creativo y de innovación, pero permitiendo la creación simultánea de productos de mayor madurez, y con más posibilidades de alcanzar un alto nivel de adopción.

Por otro lado, tanto la modularidad del software libre, como la convivencia con sistemas propietarios pueden generar serios problemas de compatibilidad, que necesitarán un trabajo de integración cuidadoso. La generalización de estándares será positiva para minimizar los aspectos adversos de la combinación de distintos elementos de software.

Certificación técnica

Las características inherentes al acabado del software libre también abren la oportunidad para la certificación por parte de integradores y consultores externos. Puede tomar dos formas, la certificación de someterse a estándares internacionales, o la certificación de la idoneidad para entornos tecnológicos concretos. El certificador proporciona la seguridad de que el paquete cumple una serie de requisitos, y es legalmente responsable de su cumplimiento.

En este sentido, el certificador proporciona un intermediario responsable a un conjunto de soluciones, factor que para muchos departamentos de nuevas tecnologías de empresas consumidoras de software es fundamental. A menudo, cuando un departamento de tecnologías de la información contrata soporte y mantenimiento, no sólo está contratando un método de resolución de incidencias, sino un responsable al que imputar los problemas y fallos que puedan surgir. La decisión de adoptar una solución concreta de software libre sin intermediarios responsables que ofrezcan garantías pone todo el peso del éxito o del fracaso sobre el propio departamento, el cual puede preferir que esta responsabilidad la asuma el intermediario.

Lectura complementaria

S. Sieber; J. Valor (2005). *Criterios de adopción de las tecnologías de información y comunicación*. IESE.
<www.iese.edu/en/files/6_15211.pdf>

Formación

La formación puede suponer una fuente de ingresos fácil. Además de que el modelo de desarrollo abierto pone a disposición de cualquiera toda la información disponible de un producto, la mayoría de los proyectos de software libre carecen de programas de formación oficiales, permitiendo a cualquiera entrar en el negocio. Muchas empresas establecidas dedicadas a la formación ya han incorporado programas de software libre entre sus ofertas.

Soporte y mantenimiento

Hemos visto ya cómo los servicios de soporte y mantenimiento suponen una fuente de ingresos importante para empresas dedicadas al desarrollo de un producto libre, pero también formarán parte de la oferta de empresas orientadas sólo a la prestación de servicios de forma horizontal, que veremos a continuación.

Como hemos dicho, el rango de empresas de servicios posible es enorme, desarrollando modelos basados en la especialización sobre ciertos servicios, sobre un tipo de aplicaciones, de manera local o a gran escala, etc. Vamos a elegir tres tipologías para estudiar en detalle. Por un lado, las empresas distribui-

doras de plataformas, por ser uno de los primeros modelos de negocio implementados con software libre, y por ser bastante representativo de un número de importantes empresas en el sector. Por otro, hemos escogido dos ejemplos que se sitúan en los extremos en cuanto a escala: las grandes integradoras, y las pequeñas microempresas de nicho. Entre ambas se situarán el resto de los modelos de negocio posibles centrados en la prestación de servicios.

3.2.1. Empresas distribuidoras de plataformas

La actividad de este tipo de empresas se centra en la integración y selección de componentes para generar una **solución de software integral**. La diversidad de aplicaciones y resultados que genera el modelo de desarrollo del software libre hace necesario el trabajo de equipos integradores que den cohesión y aseguren la compatibilidad entre las partes, lo que ha generado la aparición de distintas distribuciones, desarrolladas por distintos actores. Sin duda, esta actividad también representa un modelo de negocio potencial.

Las empresas distribuidoras de plataformas siguen un modelo similar al de las empresas desarrolladoras de una aplicación y prestadoras de servicios, pero el núcleo de su trabajo es la **selección e integración de una base amplia de productos**, en lugar del desarrollo.

> Las empresas de este modelo se dedican a generar y distribuir conjuntos integrados de software teniendo en cuenta, principalmente, al cliente corporativo. La plataforma generada constituye el producto principal de la empresa, lo que genera un problema importante: la diferenciación del producto se hace muy difícil, ya que es accesible de forma libre.

Además de distribuir software según un modelo tradicional vendiendo CD empaquetados, estas empresas suelen completar su oferta con servicios como instalación y soporte de calidad, a menudo mediante un sistema de subscripciones. El valor añadido que ofrecen se basa en la fiabilidad y en la confianza, transmitida por la marca que les representa. Ofrecen rellenar las carencias que puede presentar un producto de software libre en un entorno corporativo, en el que se busca una solución adecuada, estable y fiable, aun a costa de prestaciones y rendimiento.

Así, sus clientes potenciales serán empresas medianas y grandes, que requieran madurez y estabilidad, existencia de un soporte profesional, y un ecosistema de soluciones viable. La inversión en aplicaciones informáticas se amortiza en cinco años, por lo que una empresa que vaya a invertir en software necesitará saber que al menos durante ese tiempo, va a poder contar con soporte sobre

esos productos. Dados los costes adicionales que se generan al cambiar una solución tecnológica por otra, la duración del soporte más allá del periodo de amortización será muy deseable.

En este sentido, la generación de confianza es fundamental en su estrategia de empresa, y debe incluir el desarrollo de una marca que transmita fiabilidad añadida a un producto de software libre. Al basar su modelo de negocio en un producto accesible a cualquiera de forma libre. Estas empresas buscan como factor diferenciador el desarrollo de una marca fuerte que les permita ganar cuotas de mercado frente a productos iguales o muy similares.

Aunque estas empresas no se suelen centrar en el desarrollo de aplicaciones específicas, sí que suelen contribuir a los proyectos sobre los que se nutren corrigiendo errores, y sólo elaboran productos nuevos cuando es necesario para ampliar el mercado de su producto.

Red Hat

El ejemplo arquetípico de empresa distribuidora de plataformas es Red Hat, Inc. siendo también el modelo que sigue Novell con SuSE, Canonical con Ubuntu, y que siguió Caldera Systems con Caldera Linux.

Nuevos distribuidores

Recientemente, han aparecido nuevas compañías distribuidoras, ofreciendo conjuntos de software más especializado, para mercados más limitados. SourceLabs, SpikeSource, y Wild Open Source son ejemplos de este tipo de iniciativas. SourceLabs, por ejemplo, proporciona colecciones certificadas de software que se suele usar de forma conjunta, como Linux, Apache, PHP y MySQL. Por otro lado, Wild Open Source personaliza distribuciones para su uso en contextos de alto rendimiento, o de sistemas embebidos. Junto con el conjunto certificado, al igual que las compañías clásicas de subscripciones, las empresas ofrecen servicios de soporte y mantenimiento en torno a su selección.

El principal reto de este tipo de empresas será lograr definir colecciones de software suficientemente amplias como para mantener una base suficiente de clientes, pero ser capaces de ofrecer soporte sobre todos los elementos del conjunto.

El caso de SpikeSource

Nombre de la compañía	Spikesource, Inc.
Sede	Redwood City, CA. (Estados Unidos)
Página web	www.spikesource.com
Fecha de creación	2003
Nº de personas empleadas en 2006	80
Volumen de ventas en 2007 (millones)	

Datos corporativos de SpikeSource, Inc. Elaborada a partir de estadísticas de Hoovers (http://www.hoovers.com)

SpikeSource supone un claro ejemplo del potencial de negocio tras la falta de acabado de los productos de software libre. Nació en el 2003, en el seno de una de las firmas de capital riesgo más importantes del boom de Internet, Kleiner Perkins Caufield & Byers, y lanzó sus primeros productos en abril del 2005. En octubre del 2006 la empresa anunció su expansión en Europa a través de una red de proveedores de soluciones locales y socios tecnológicos.

Murugan Pal, fundador de la empresa, resume la actividad de la compañía de la siguiente manera:

"La meta de SpikeSource es facilitar la adopción de software de código abierto en la empresa a través de pruebas, certificación y servicios de soporte. Nosotros innovamos, automatizamos y optimizamos técnicas de testeo avanzadas como parte de nuestra actividad principal".

(Murugan Pal. "Participatory Testing: The SpikeSource Approach". http://www.oreillynet.com/pub/a/network/2005/04/07/spikesource.html)

Como factores diferenciadores frente a otros distribuidores de soluciones integradas clásicas, como Red Hat, la empresa destaca sus esfuerzos por promover la automatización de las pruebas, así como la combinación de aplicaciones específicas, que puedan instalarse sobre distintas plataformas y sistemas operativos. Incluye versiones para diversos sistemas operativos, tanto libres como propietarios, además de incluir software cerrado en algunos de los productos.

Además de los conjuntos que integra, como SpikeWAMP-1.4, que proporciona las últimas versiones de PHP, MySQL y Apache, para ser instalado sobre Windows, y "SuiteTwo", orientado a proporcionar una amplia gama de aplicaciones integradas para la colaboración, y funcionalidades "Web 2.0", recientemente ha lanzado una plataforma para desarrolladores, sobre la que éstos pueden probar e integrar sus aplicaciones, obteniendo así la certificación de Spikesource, y consiguiendo como resultado un mayor acabado (productización) de su software.

El trabajo de este tipo de empresas puede resultar muy favorable a la hora de dar visibilidad y fomentar la adopción de soluciones libres, y SpikeSource destaca esta labor. La empresa se esfuerza por demostrar que su trabajo beneficia a la comunidad de software libre, y no sólo la explota, incluyendo figuras reconocidas del mundo del software libre en su comisión asesora, como Brian Behlendorf y Larry Rosen para avalarlo. También ha desarrollado una web orientada a desarrolladores (http://developer.spikesource.com), desde la que ofrece sus servicios de pruebas automáticas de integración y compatibilidad sobre distintas plataformas.

Sin embargo, el **software de automatización** con el que trabaja la empresa combina partes que han sido liberadas, con partes que permanecen cerradas. Reservarse una porción del código supone en este caso una estrategia para proteger su diferenciación, y evitar la entrada de competencia con servicios equivalentes. Se pone de manifiesto con esta elección que más que la pérdida de ingresos por licencias (que como hemos visto a lo largo de la asignatura no supone un obstáculo real), el uso de software libre afectará a las posibilidades de diferenciación, y por lo tanto de negocio, que tenga la empresa. En el caso de SpikeSource, el trabajo invertido en sus aplicaciones de pruebas se verá recompensado no por la venta de licencias sobre ese software, sino por la protección de su factor diferenciador frente a otras empresas ofreciendo servicios similares.

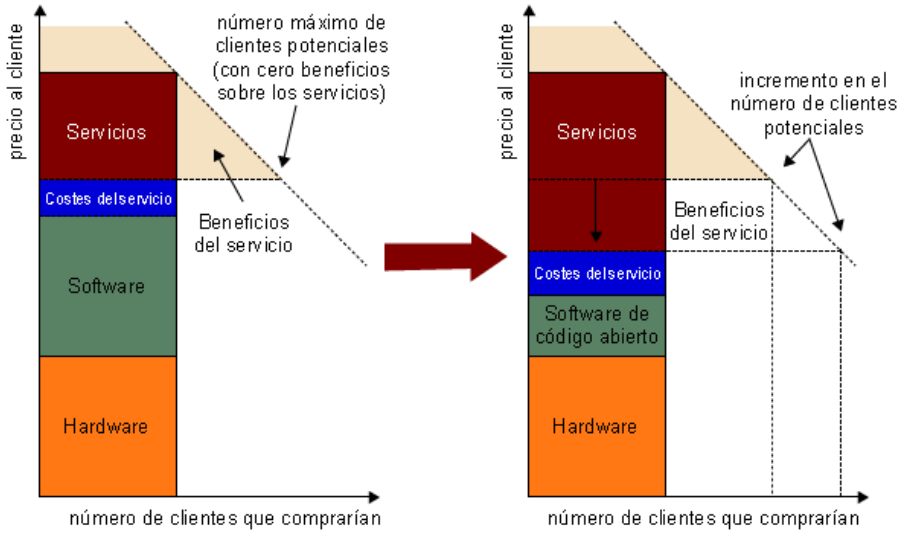
3.2.2. Grandes integradoras

Las grandes integradoras de sistemas, o generadores de soluciones, son unos de los tipos de empresas que más pueden ganar basando su negocio en el uso de software libre, debido al ahorro de costes directo, y la posibilidad consecuente de alcanzar más clientes.

Normalmente, un cliente estará buscando una empresa que le proporcione soluciones a un problema de tecnologías de la información y comunicación (TIC), sin importarle los detalles de implementación. Una solución completa combinará hardware, software y servicios, facilitando el proceso al cliente, que sólo necesitará contactar con una empresa para resolver sus problemas TIC, sin tener que preocuparse por la compatibilidad entre proveedores. Por lo tanto, todo lo que la empresa pueda ahorrar en costes de software empleando software libre lo podrá trasladar a los costes de servicios, mejorando así su solución. Podrá bajar el precio, aumentando así su número potencial de clientes, o

simplemente podrá mejorar su rentabilidad. Este tipo de grandes integradores que, en general, trabajan en proyectos complejos, podrán mantener los precios dadas las dificultades de entrada para otros competidores.

El siguiente gráfico ilustra esta situación, dibujando la curva de demanda de soluciones integrales, y los costes del proveedor.



Curva de la demanda de servicios informáticos integrales. Márgenes de ventas y número de clientes.
Fuente: Dirk Riehle, "The Economic Motivation of Open Source Software: Stakeholder perspectives".
http://www.riehle.org/computer-science/research/2007/computer-2007-article.html)

Existen multitud de empresas de consultoría y selección como Ayamon, Enomaly, Navica, OpenLogic, Optaros, X-tend. Como grandes integradoras, podemos destacar a IBM, Sun, y HP.

El caso de IBM

Nombre de la compañía	IBM
Sede	Armonk, NY (Estados Unidos)
Página web	www.ibm.com
Fecha de creación	Su origen se remonta a 1896. En 1924 toma el nombre de IBM
Nº de personas empleadas en 2007	394.540
Ingresos en 2007 (millones)	$91.423

Datos corporativos de IBM. Elaborada a partir de estadísticas de SoftwareMagazine (www.softwaremag.com) y Wikipedia.

Hace veinte años, **IBM** era uno de los más fuertes defensores de los derechos de propiedad intelectual para el software. Defendía que sin una protección fuerte de *copyright*, no habría incentivos para que las empresas invirtieran en el desarrollo de software.

Actualmente, aunque mantiene el grueso de su software propietario, IBM ha puesto en marcha importantes campañas de apoyo al software libre, con considerables contribucio-

nes económicas al desarrollo de Linux y otras aplicaciones, y la liberación de aplicaciones como la plataforma de desarrollo Eclipse, y parte de su sistema operativo AIX.

El modelo de negocio actual de IBM se enfoca en la **venta de hardware de alta gama**, software propietario sobre Linux, y la **prestación de servicios de integración a clientes corporativos**. Aunque IBM ha sido uno de los principales fabricantes de software a nivel mundial, sus programas han sido normalmente comercializados como una solución combinada con su propio hardware. En este sentido, la empresa no tiene mucho que perder por la falta de diferenciación del software que usa: dadas las dificultades de entrada a competidores en el campo de los *mainframes,* el uso de software de bajo coste permitirá a la empresa bajar sus precios y ampliar su gama de clientes, pero no sufrirá una pérdida de diferenciación que aumente de forma significativa la competencia.

De esta manera, el uso de Linux ha permitido a IBM ofrecer un precio global más bajo por su hardware y sus servicios, pero también ha proporcionado una plataforma común sobre la que construir y vender aplicaciones y servicios especiales. En este sentido, cabe destacar también el ahorro que supone para la empresa el uso de un sistema operativo con gran adopción previa, en cuanto a marketing, difusión y ventas, además de la reducción del riesgo y la inversión en desarrollo. Los beneficios obtenidos en cuanto a imagen pública, por otro lado, resultan asimismo significativos.

Desde luego, la actividad de IBM en torno al software libre responde a una estrategia más compleja, que le permite obtener una mejor posición competitiva en diversos frentes. Desde las estrategias basadas en el uso de software libre para mejorar la comercialización de sus productos propietarios (como los "*loss leader*" y núcleo libre más accesorios propietarios), a conseguir una mejor posición respecto de otras grandes empresas proveedoras de software.

El uso de software libre ha permitido a IBM obtener más independencia respecto a otras grandes empresas como Microsoft, y una mejor posición respecto a competidores directos como Sun. Esta última ha basado su estrategia de negocio durante mucho tiempo en la venta combinada de hardware más sistemas operativos "mejores que la media", por lo que tendría más que perder frente a la reducción de costes y la presencia de software equivalente de bajo precio.

3.2.3. Servicios en torno al software: pequeñas y microempresas

Otro fenómeno fundamental que desencadena el software libre es el de la **transferencia tecnológica y de conocimiento**. La inversión en formación, desarrollo y tecnología, tanto a escala de grandes empresas como a escala individual, se hace disponible a través de desarrollos abiertos a cualquiera con una conexión a Internet, y con ciertos conocimientos.

Este fenómeno puede tener un gran impacto a nivel de transferencia tecnológica tanto entre países más y menos desarrollados, como de forma interna, entre grandes empresas multinacionales y microempresas locales.

La posibilidad de acceder tanto al código como a las decisiones de diseño y desarrollo libremente supone un gran potencial para pequeñas empresas tecnológicas, que pueden estar en contacto y hacer suya la tecnología más innovadora apoyada por una gran inversión financiera.

Dado su tamaño, por lo general estas empresas orientarán su actividad hacia nichos concretos, necesitando un número reducido de clientes para mantener el negocio. Las posibilidades de segmentación de mercados son múltiples,

aunque un factor común podrá ser el de proporcionar un trato más cercano y personalizado (muchos clientes preferirán ser los clientes principales de una pequeña empresa, que uno más sin importancia de una gran multinacional).

Entre este tipo de empresas, es interesante destacar aquellas que basan su diferenciación en emplear software libre no sólo por las ventajas que hemos visto hasta ahora, sino como una **declaración de intenciones**, como un elemento más dentro de una lógica empresarial que no busca la acumulación de beneficios, sino la generación de medios de vida autosustentados, a través de la prestación de servicios que contribuyan al desarrollo y bienestar de la sociedad.

El funcionamiento interno de estas empresas a menudo refleja también esta filosofía y aproximación al mundo empresarial, basándose en la **horizontalidad** y la **transparencia**. Es interesante destacar que el marco jurídico español ofrece una figura empresarial bastante ajustada a lo que hemos descrito: las cooperativas de trabajo, en las que no existen socios capitalistas, y son los propios trabajadores los que dirigen y controlan la empresa.

De nuevo, el concepto de la **ética empresarial** no es nuevo ni exclusivo del software libre, pero cobra un especial significado en este tipo de empresas. A menudo, estas pequeñas empresas se agrupan mediante distintos tipos de redes, lo que supone una estrategia fundamental para promover el apoyo y la colaboración entre ellas, siguiendo los principios éticos y políticos sobre los que se asientan.

Una parte importante de los clientes potenciales serán otras empresas con similares principios de funcionamiento, organizaciones con una motivación social o política, y administraciones públicas.

Como ejemplos de este modelo, varias empresas españolas con un tipo de funcionamiento similar se han estado uniendo en torno al grupo Ikusnet (http://www.grupoikusnet.com/), bajo los siguientes principios:

> "Nuestra base metodológica se asienta en la cooperación y la 'horizontalidad' para la toma y aplicación de decisiones, de forma que el propio modo de cooperación se convierte en fuerza productiva que pretende desplegar sus efectos en el marco de la sociedad de la información y el conocimiento."

También podemos destacar la cooperativa madrileña **Xsto.info** (http://xsto.info), una microempresa de menos de diez trabajadores. Nacida en el seno de movimientos sociales, se constituyó como cooperativa de trabajo en el año 2003. Esta elección de forma jurídica supone, al igual que el uso de software libre, una declaración de intenciones en cuanto a sus principios de funcionamiento, que completan en su página web, con el siguiente lema:

> "Aún estamos a tiempo de participar de esta transformación social para que se produzca de una forma participativa, abierta, libre y democrática".

Cuenta entre sus clientes con administraciones locales como el Ayuntamiento de Parla, y distintos tipos de asociaciones, como la Federación Regional de Asociaciones de Vecinos de Madrid.

Otro ejemplo muy representativo, y de especial interés por su edad, es el caso de Easter-eggs que examinamos a continuación.

El caso de Easter-eggs

Nombre de la compañía	Easter-eggs
Sede	París, Francia
Página web	www.easter-eggs.com
Fecha de creación	1997
Nº de personas empleadas en 2007	15
Cifra de negocio en 2006	800.000€

Datos corporativos de Easter-eggs. Obtenidos de su página web (http://www.easter-eggs.com)

Easter-eggs es una pyme francesa dedicada a prestar servicios en torno al software libre, con una consolidada trayectoria. Creada en 1997, ofrece un amplio abanico de servicios, desde instalación, administración y seguridad de sistemas GNU/Linux; adaptación de aplicaciones y desarrollos a medida; a consultoría, auditoría y formación. Es una de las empresas de servicios en torno al software libre más antiguas, y goza de buena salud: rentable desde el momento de su formación, actualmente cuenta con quince personas asalariadas, y en el 2006 alcanzó una cifra de negocio de 800 mil euros. Entre sus clientes, podemos mencionar a la Universidad René Descartes de París (http://www.univ-paris5.fr/), y Europcar, para la que pusieron en marcha un programa de migración a GNU/Linux en 3.500 de sus agencias.

Para la empresa, la elección de prestar servicios en torno al software libre se basó sobre principios éticos más que financieros, y estos principios son los que la han llevado también a definir un esquema de funcionamiento empresarial muy particular. De una manera muy similar a la que rige el funcionamiento de las cooperativas de trabajo españolas, la empresa Easter-eggs está totalmente controlada por sus personas asalariadas, y sólo por ellas. No hay socios capitalistas, ni ningún tipo de inversión externa. Para conseguir este esquema organizativo, crearon una asociación, la Asociación de Asalariados de Easter-eggs (http://www.easter-eggs.org), que posee el 99,8% del capital de la empresa.

Sobre esta base, Easter-eggs construye su diferenciación empresarial, denominándose a sí mima como "una empresa social" con una preocupación central por crear una "empresa ciudadana", que dé respuesta a las aspiraciones crecientes de los ciudadanos que comienzan a percibir los límites del consumismo, y reclaman un comportamiento con sentido de las empresas. Entre sus principios de funcionamiento destacan la transparencia financiera (su contabilidad está disponible para la descarga desde su página web: http://www.easter-eggs.org/rubrique_10_Comptabilite.html), la igualdad salarial, y mecanismos de implicación y corresponsabilidad por parte de sus trabajadores.

Como parte de la estrategia de creación de redes, y de agrupación de pequeñas empresas socialmente responsables, para poder prestar servicios a gran escala, y como medio de promoción conjunto, la asociación Easter-eggs creó, en el año 2002, la red *libre enterprise* (http://www.libre-entreprise.org), en la que se pueden encontrar unas dieciséis empresas francesas de servicios basados en software libre, siguiendo similares modelos de negocio.

3.3. Mercados auxiliares: Hardware

Uno de los primeros modelos de negocio descritos por Hecker, "Widget Frosting", sigue igual de vigente en la actualidad. Para fabricantes de hardware, el desarrollo de software supone un gasto necesario para poder vender sus pro-

ductos, por lo que cualquier estrategia que permita bajar los costes relacionados es deseable. Además, seguir un modelo de desarrollo de software libre amplia las posibilidades de portabilidad a otras plataformas, aumentando su segmento de mercado. Hemos visto anteriormente cómo las grandes proveedoras, que incluyen hardware en su oferta, están incluyendo sistemas operativos libres como una manera de reducir los costes finales del servicio, y por lo tanto aumentar su base de clientes potenciales.

En este campo, es interesante destacar el papel que está teniendo Linux en la nueva generación de **dispositivos empotrados**. Hemos vivido un retorno a la venta conjunta de hardware y software en este tipo de dispositivos, que deben venderse con su funcionalidad concreta incorporada, a menudo con sistemas operativos simples y de escasa funcionalidad. Sin embargo, la posibilidad de usar Linux empotrado ha multiplicado las oportunidades de negocio de este tipo de hardware.

El uso de software libre ofrece importantes ventajas en términos de ahorro de costes, de acortamiento de los periodos de desarrollo (fundamental en un mercado que se rige por ciclos de vida cortos), de facilidad a la hora de subcontratar desarrollos (al partir de una base existente de gran modularidad), y las posibilidades de innovación que brinda construir una comunidad en torno al producto. Por otra parte, el uso de software libre otorga a los fabricantes una importante independencia frente a las plataformas Windows Mobile y Symbian, y por lo tanto, de las agendas de Microsoft y Nokia.

Actualmente, los sistemas operativos basados en Linux son los más utilizados en sistemas empotrados, y su adopción por parte de empresas consolidadas en el sector, como Wind River, avala la continuidad de esta tendencia. En el mercado de los móviles SmartPhones, Linux pasó del 3,4% en el 2004 al 14,3% en el 2005, mientras que Windows empotrado sólo creció del 2,9% al 4,5% en el mismo periodo.

Web recomendada

Para más información:
Alejandro Lucero, "Seminario UAM: Linux en Sistemas Empotrados". *www.os3sl.com/Documents/Seminario_UAM_I.pdf*.

Por otro lado, la existencia de software a un precio asequible para una extensa audiencia también genera un ecosistema de necesidades a su alrededor, del que muchas veces forma parte el hardware. La plataforma de voz por IP **Asterisk**, por ejemplo, pone al alcance de muchos negocios el establecimiento de centralitas telefónicas, y una importante reducción de gastos. Sin embargo, su implementación lleva consigo la necesidad de adquirir determinados elementos de hardware, como terminales IP, tarjetas Asterisk, *routers*, sistemas de grabación, etc.

Los fabricantes de estos productos se podrán beneficiar de la difusión de software como Asterisk, por lo que tendrán mucho que ganar de la participación y contribución en su desarrollo. De manera inversa, las empresas desarrolla-

doras del software podrán obtener ingresos mediante la venta de hardware y servicios relacionados, tal y como hace Digium, responsable principal del desarrollo de Asterisk.

Pero además, surgen otros espacios y nichos susceptibles de ser explotados gracias a esta tecnología, como los que aprovecha **Avanzada7**. Esta empresa malagueña vende el hardware necesario para la implementación de Asterisk, pero reconoce que no es un fabricante, ni un gran distribuidor. Su aspecto diferenciador es la oferta de servicios de soporte gratuitos a partir de la venta de los dispositivos. Además, Avanzada7 ha establecido relaciones con Digium, la empresa responsable del desarrollo del software, generando una red de confianza que extiende hacia otras empresas que deseen implementar Asterisk en clientes finales. Pone en marcha, de esta manera, una red del tipo piramidal descrito anteriormente, basándose en necesidades generadas por un software libre, y explotándolas a través de estrategias de *coopetencia*.

El caso de Chumby

Nombre de la compañía	Chumby Industries, Inc.
Sede	San Diego, CA (Estados Unidos)
Página web	www.chumby.com
Fecha de creación	2005
Nº de personas empleadas en 2007	
Cifra de negocio en 2006	

Datos corporativos de Chumby Industries, Inc.

Chumby Industries se creó con el objetivo de crear y comercializar "el Chumby", lanzado en agosto del 2006. Se trata de un dispositivo inalámbrico (wifi), orientado a reemplazar la radio-despertador, con capacidad de conexión a la "red Chumby", desde donde descargar distintos tipos de información. Puede reproducir *podcasts,* radio por Internet, y algunos vídeos. El dispositivo funciona con Linux, y con Flash Lite, programa de Adobe para proporcionar pequeñas aplicaciones interactivas: "*widgets*". No cuenta con un navegador, el contenido sólo se puede descargar a través de los *widgets,* diseñado cada uno para una función concreta: leer las últimas noticias de un blog, descargar las últimas fotos de una galería, etc.

Tanto el software como el hardware de Chumby son libres, se pueden descargar tanto sus esquemáticos y circuitos impresos, como su código fuente. La empresa centra su actividad de marketing en torno a su apertura: el Chumby se puede personalizar a todos los niveles, cambiándole la carcasa exterior y cosiéndole (literalmente) extensiones al gusto, crear nuevos *widgets,* o "hackear" el hardware. De esta manera, el dispositivo no sólo se vende como "amigo del usuario", sino que abre la puerta a la expansión de sus funcionalidades más allá del control y financiación de la empresa, dejando que evolucione hacia lo que cada usuario quiera que sea.

Sin embargo, el modelo de negocio de Chumby no es obtener ingresos por hardware, y el precio del dispositivo se mantiene relativamente bajo. Steve Tomlin, fundador y gerente de la compañía argumenta que varios modelos de negocio se hacían posibles con Chumby: podrían cobrar más y seguir un modelo de vendedor de hardware tradicional, con los problemas de ingresos recurrentes que se generarían, o podrían cobrar poco por el dispositivo, pero cobrar subscripciones por el contenido. Sin embargo, prefirieron una tercera vía: obtener los ingresos justos para cubrir costes con las ventas, y obtener beneficios a través de la publicidad.

Para garantizar este nuevo campo de negocio, Chumby no es 100% abierto, estableciendo limitaciones en cuanto a su uso, tanto a nivel de hardware como a nivel de la "red Chumby", para garantizar su modelo de negocio.

Acceso a la "red Chumby"

Tras adquirir un Chumby, el usuario debe registrarse en la web de la empresa para tener acceso a los *widgets*, aceptando unos términos de uso. Estos términos permiten que cualquiera incorpore nuevos *widgets*, con cualquier tipo de información que deseen, dando permiso para su distribución a cualquier dispositivo conectado a la red. Se establecen, sin embargo, unos límites en cuanto al contenido permitido, prohibiendo contenido inadecuado (racista, violento, sexista, *spam*...), pero también contenido comercial:

"**Prohibited Content** includes Content that: (...) except as expressly approved by Chumby, involves commercial activities and/or promotions such as, without limitation, contests, sweepstakes, barter, advertising, or pyramid schemes." (http://www.chumby.com/pages/terms)

Los contenidos publicitarios, por lo tanto, necesitarán pagar para obtener la autorización. Por otro lado, los términos también avisan de que al conectarse a la red Chumby, se recibirá publicidad.

Aunque la incorporación de *widgets* al margen de la red Chumby es técnicamente posible a través de dispositivos USB, la empresa confía en que la mayor parte de las aportaciones se mantendrán en el seno de su red, consiguiendo atraer suficiente contenido como para cobrar valor por el número de personas y de contribuciones en torno a ella.

El dispositivo

Chumby permite el acceso a los esquemáticos y PCB de su dispositivo, sin embargo, los fabricantes que quieran aprovechar sus diseños e incorporarlos a sus productos, tendrán que pagar a la empresa por licenciar su nuevo producto, y tendrán que aceptar que, además de otras redes a las que se conecte, también cuente con la red Chumby.

En resumen, Chumby reconoce que el valor de su dispositivo está en el contenido, de una manera análoga a la de O'Reilly en "Open Source Paradigm Shift", y otros. Su estrategia, además de caracterizar a su producto por su apertura, es atraer al mayor número de personas a su red, intentando convertirla en red de referencia para pequeños dispositivos móviles de este tipo. Sin embargo, en lugar de vender el contenido a través de subscripciones, ha decidido capitalizar el valor a través de la publicidad.

Para la empresa, el uso de software y hardware abierto supone una estrategia clave de cara a la difusión y adopción, no sólo de su dispositivo, sino de la red que ha creado para proporcionarle contenido. Por otra parte, su carácter abierto le proporciona una clara diferenciación y ventaja comercial frente a productos similares como el iPod Touch y el iPhone de Apple.

3.4. Otros mercados auxiliares

La propia generalización del software libre, tanto por su forma de desarrollo como por su uso, genera en sí misma otros mercados relacionados que han sido explotados por distintas empresas:

- **Comunidad y desarrollo:** quizás los ejemplos más obvios sean los que proporcionan servicios de alojamiento y herramientas de trabajo colaborativo para proyectos de software, como SourceForge, Collabnet, o FreshMeat. También podemos destacar la proliferación de buscadores de código como Google Code, Koders, Krugle o Codase.

- **Certificaciones legales:** también crecen en importancia las empresas dedicadas a este tipo de certificaciones, que aseguren que un software o una combinación concreta será posible legalmente, y conozca los problemas derivados de las licencias que puede tener. Este servicio lo están ofreciendo

empresas que hemos visto antes, como las creadoras de plataformas y "paquetes", como SpikeSource, pero también han surgido algunas dedicadas íntegramente al trabajo legal, como Black Duck, y Palamida.

- **Venta de libros:** O'Reilly con sus libros es otro de los ejemplos más mencionados de estas categorías.

- ***Merchandising:*** tampoco debemos olvidar la importancia del *merchandising* como forma de financiación complementaria, o incluso principal. Podemos destacar el papel que juega ThinkGeek, filial de SourceForge, en la aportación de ingresos, a través de la venta por Internet de diversos tipos de productos para *geeks*: desde camisetas y tazas de té, hasta diversos dispositivos o *gadgets*.

Resumen

Este módulo se ha dedicado a estudiar detalladamente los diferentes modelos de negocio válidos y viables basados en el software libre. El crecimiento de empresas dedicadas íntegramente a su explotación, así como la reorientación de la estrategia de multinacionales del software, es una prueba concluyente.

En un primer momento, se han presentado diferentes clasificaciones propuestas por distintos autores a lo largo del tiempo:

- Las clasificaciones de Hecker y de Raymond, basadas en la observación de empresas que usaban el software libre como parte de sus modelos de negocio.

- La clasificación del Grupo de Trabajo Europeo de Software Libre, basada en el modelo de financiación empresarial.

- La clasificación de Daffara, basada en estudios empíricos.

Finalmente, se ha propuesto y desarrollado detalladamente una propuesta propia y particular de modelos de negocio:

- Especialistas/verticales, centradas principalmente en el producto de software libre y que eventualmente pueden adoptar modelos mixtos de doble licencia, de accesorios propietarios, de venta distribuida del producto, o bien modelos de comercialización de servicios sobre el producto, como el software como servicio.

- Servicios asociados, como los desarrollos a medida, la selección, instalación, integración, certificación técnica, formación, soporte y mantenimiento de productos; que eventualmente se pueden orientar a la distribución de plataformas, a la integración a gran escala o al servicio de pequeñas empresas y microempresas.

- Mercados auxiliares de hardware, que con el software libre complementan una orientación principal de negocio a la venta de productos físicos o bien directamente al negocio de los contenidos accesibles desde un determinado hardware.

- Otros mercados auxiliares, como las herramientas de colaboración, las certificaciones legales, la venta de libros o el *merchandising*.

Bibliografía

Augustin, L. (2007). "A New Breed of P&L: the Open Source Business Financial Model". Open Source Business Conference (OSBC)*Metcalfe, Randy*. <http://www.osbc.com/live/images/13/presentation_dwn/A_New_Breed_of_P_and_L.pdf> [Consulta: febrero 2009]

Mickos, M. (2007). "Open Source: why freedom makes a better business model". *Open Source Business Conference (OSBC)*. <http://www.osbc.com/live/images/13/presentation_dwn/Keynote-Open_Source_Why_Freedom.pdf> [Consulta: junio 2008]

West, J. y Gallagher, S. (2006). "Patterns of Open Innovation in Open Source Software". En: Henry Chesbrough; Wim Vanhaverbeke; Joel West (eds.). *Open Innovation: Researching a New Paradigm* (pág. 82-106). Oxford: Oxford University Press. <http://www.openinnovation.net/Book/NewParadigm/Chapters/index.html> [Consulta: junio 2008].

Capiobanco, F.; Onetti, A. (julio, 2005). "Open Source and Business Model Innovation. The Funambol case". En: M. Scotto; G. Succi (eds.). *Proceedings of First International Conference on Open Source* (OSS2005) (pág. 224-227). Génova. <http://oss2005.case.unibz.it/Papers/4.pdf> [Consulta: junio 2008].

Riehle, D. (2007). "The Economic Motivation of Open Source Software: Stakeholder perspectives". *IEEE Computer* (vol. 4, núm. 40, págs. 25-32). <http://www.riehle.org/computer-science/research/2007/computer-2007-article.html> [Consulta: febrero 2009]

Comino, S.; Manetti, F. M. (2007). *Dual licensing in open source markets*. Univerista Degli Studi di Trento, Departamento de economía. <http://www-econo.economia.unitn.it/new/pubblicazioni/papers/18_07_comino.pdf> [Consulta: junio 2008].

Daffara, C. (2007). *Business models in FLOSS-based companies*. Conecta Reasearch, 2007. <http://opensource.mit.edu/papers/OSSEMP07-daffara.pdf> [Consulta: junio 2008]

Pal, M. (julio, 2005). "Participatory Testing: The SpikeSource Approach". O'Reilly Network. <http://www.oreillynet.com/pub/a/network/2005/04/07/spikesource.html> [Consulta: junio 2008]

Kelsey, J.; Schneier, B. (junio, 1999). "The Street Performer Protocol and Digital Copyrights". *First Monday* (vol. 4, núm. 6). <http://www.firstmonday.dk/issues/issue4_6/kelsey/> [Consulta: junio 2008]

Rasch, C. (junio, 2001). "The Wall Street Performer Protocol". *First Monday* (vol. 6, núm. 6). <http://www.firstmonday.org/issues/issue6_6/rasch/index.html> [Consulta: junio 2008]

Daffara, C.; Barahona, J. B. y otros (2000). "Free Software/Open Source: Information Society Opportunities for Europe?" *Working paper*. <http://eu.conecta.it/paper/> [Consulta: febrero 2009]

Lucero, Alejandro. "Seminario UAM: Linux en Sistemas Empotrados". <www.os3sl.com/Documents/Seminario_UAM_I.pdf> [Consulta: Junio 2008]

Raymond, E. (1999). *The Magic Cauldron* <http://catb.org/~esr/writings/magic-cauldron/>

Traducción al español en: <http://gnuwin.epfl.ch/articles/es/magiccauldron/es-magic-cauldron/es-magic-cauldron.html> [Consulta: febrero 2009]

Hecker, F. (1998). *Setting Up Shop. The Business of Open Source Business* <http://hecker.org/writings/setting-up-shop> [Consulta: febrero 2009]

Metcalfe, Randy (2006). *Open Source Business: Differentiation and Success* <http://www.oss-watch.ac.uk/resources/businessmodels.xml> [Consulta: febrero 2009]

Casos de estudio

50 Open Source Success Stories in Business, Education, and Government <http://www.blogcrm.com/50-open-source-success-stories-in-business-education-and-government.php>

Red Hat and J. Boss. "Is Open Source viable in Industry? The case of Red Hat and JBoss". <http://www.whyfloss.com/es/conference/madrid08/getpdf/68>

Avanzada7. "Business models based on Asterisk: The case of Avanzada7". <http://www.whyfloss.com/es/conference/madrid08/getpdf/64>

Openbravo. "Openbravo: keys to success in free software application development". <http://www.whyfloss.com/es/conference/madrid08/getpdf/49>

Liferay. "Liferay Enterprise Portal: The project, the product, the community and how to extend it". <http://www.whyfloss.com/es/conference/madrid08/getpdf/66>

Varios casos. <http://www.opensourceacademy.gov.uk/solutions/casestudies>

Desarrollar software libre en una empresa

Amadeu Albós Raya

PID_00145047

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

En este módulo nos adentramos en el mundo de la producción de software libre y en sus particularidades más relevantes para el producto, la empresa y la comunidad de usuarios.

En un primer momento, analizaremos el desarrollo de software libre bajo el punto de vista del proyecto, considerando los principales aspectos ligados a la población y la gestión del proyecto, así como la participación de la comunidad de usuarios en diversidad de aspectos.

Mediante el proyecto de software libre se formaliza la relación entre empresa y comunidad de usuarios. La adecuación de las particularidades de dicha relación es fundamental para la consecución de los objetivos del proyecto.

A continuación, describiremos las particularidades de la comunidad de usuarios del software libre y su gestión por parte de la empresa. Esta gestión complementa la metodología de producción e implementa la estrategia relacional comentada anteriormente.

Finalmente, el módulo concluirá con la presentación de un caso de estudio de una empresa real que produce software libre.

Este módulo se organiza como una guía de lecturas externas, cuyo objetivo es profundizar en las particularidades de los diferentes aspectos que se presentan y que son relevantes para la producción empresarial de software libre.

Objetivos

Los objetivos que se deben alcanzar al finalizar este módulo son los siguientes:

1. Familiarizarse con la metodología de producción de software libre.

2. Comprender la importancia que tiene la comunidad de usuarios en el desarrollo de productos basados en software libre.

3. Identificar y analizar los factores relevantes que afectan al éxito de la producción de software libre.

4. Entender la importancia de formalizar una metodología que permita complementar los esfuerzos de la empresa y de la comunidad de usuarios.

5. Profundizar en las implicaciones directas e indirectas de llevar a cabo un proyecto de desarrollo basado en software libre.

1. La producción de software libre

En este primer apartado nos centraremos en la producción de software libre desde el punto de vista de su desarrollo o elaboración, es decir, con independencia de eventuales modelos de negocio que lo exploten de forma lucrativa.

En diversas asignaturas del máster, especialmente en las asignaturas dedicadas a la producción del software[1], se analiza con detalle el proceso tecnológico que caracteriza la elaboración de software libre.

(1)Introducción al desarrollo de software, Ingeniería del software en entornos de software libre, o Conceptos avanzados de desarrollo de software.

Este proceso tecnológico es complementario a las metodologías que nos permiten formalizar un proyecto cooperativo viable y duradero a lo largo del tiempo. En este sentido, la colaboración de la comunidad de usuarios en el proyecto de software libre es fundamental para conseguir una masa crítica de usuarios que permita la viabilidad del proyecto.

En consecuencia, buena parte de estas metodologías y actuaciones van encaminadas a ofrecer soporte y garantías a las relaciones entre el proyecto y la comunidad de usuarios. Para darnos cuenta de la importancia de esta relación, basta con consultar los recursos que ofrecen a la comunidad de usuarios los proyectos de software libre más populares.

Proyectos populares

Por ejemplo, OpenOffice.org (http://contributing.openoffice.org/) y Mozilla (http://www.mozilla.org/contribute/).

Para desarrollar estos conceptos, en los próximos apartados describiremos tres puntos de vista complementarios. En un primer momento, consideraremos algunas ideas básicas sobre la producción de software libre. A continuación, detallaremos de forma breve los principales pasos para poner en marcha un proyecto basado en software libre. Por último, profundizaremos en los principales aspectos de gestión del proyecto de software libre.

1.1. La producción de software libre

La producción de software libre, del mismo modo que la producción de cualquier software, responde a la necesidad de solucionar una problemática tecnológica concreta[2].

(2)Por ejemplo, añadir funcionalidades a una aplicación o solucionar errores de funcionamiento.

Si bien el proceso tecnológico de refinar y hacer evolucionar una aplicación de software libre puede presentar muchas similitudes comparado con una aplicación basada en software propietario, la diferenciación que supone la aper-

tura del modelo le confiere un funcionamiento particular. Es decir, el carácter abierto y cooperativo de su producción incide en la estructura de evolución cuantitativa y cualitativa a lo largo de las versiones.

Muchos son los autores que han escrito sobre las particularidades de producir software libre. Puesto que este módulo no tiene como objetivo detallar o describir exhaustivamente dichas particularidades, dado que se tratan de forma exhaustiva en otras asignaturas, nos centraremos en remarcar algunas de las más interesantes para el caso que nos ocupa.

Para ello, consideraremos algunos de los conceptos presentes en el ensayo *La catedral y el bazar*, de Eric S. Raymond, donde se analizan las particularidades del software libre y, en especial, del caso GNU/Linux.

Web recomendada

E. Raymod (1997). *The cathedral and the bazaar* (http://www.catb.org/~esr/writings/cathedral-bazaar/).

- **El origen de la producción**
  A grandes rasgos, la producción del software libre nace a partir de las necesidades particulares del usuario o desarrollador en su actividad habitual. Es decir, la colaboración en el desarrollo del software se empieza buscando y hallando un problema cuya resolución nos resulte interesante, relevante o necesaria.

Inicios de la producción

Buena parte de los fundamentos del software libre se basa en la publicación de los ajustes o desarrollos concretos que realizan trabajadores en el ejercicio de sus funciones laborales cotidianas.

- **La comunidad de usuarios**
  La comunidad de usuarios de software libre, que agrega tanto usuarios finales como desarrolladores y programadores, es la base que da sentido a la definición de desarrollo de software libre.
  Tratar a los usuarios como colaboradores en el proyecto de producción es la forma más fácil de depurar y mejorar el código de forma rápida (si la base de colaboradores es suficiente).
  Por ende, los colaboradores representan uno de los recursos más valiosos para el desarrollo de la aplicación, por lo que también resulta valioso reconocer las buenas ideas y las soluciones que aportan.

- **Las versiones de la aplicación**
  Una de las características de la producción de software libre es la reutilización y reescritura del código original para derivarlo en un código nuevo, ya sea libre de errores, con nuevas funcionalidades o con mejor rendimiento (entre otros aspectos).
  Por otra parte, en los proyectos de desarrollo de software libre se promueve la liberación de código de forma rápida y frecuente, de forma que la actividad del proyecto sea dinámica e incesante.

- **La coordinación de la producción**
  La persona –o personas– que coordina el proyecto tiene que saber gestionar el potencial global de la comunidad de usuarios, dirigiendo la evolución del proyecto sin coerción y aprovechando los medios y sinergias que ofrece una red como Internet.

La herencia del código de la aplicación y de su gestión de coordinación es importante para el futuro del proyecto de desarrollo de software libre. La elección del sucesor que controle y gestione la producción no se debe dejar al azar.

1.2. El proyecto de software libre

Complementando las consideraciones tecnológicas y funcionales de las aplicaciones basadas en software libre, uno de los objetivos primordiales de todo proyecto de software libre es la difusión de la aplicación o la obtención de una masa crítica de usuarios.

Es decir, es poco útil para el futuro del proyecto que el código generado, a pesar de resolver problemas o carencias concretas, no sea conocido y aplicado por los usuarios potenciales. Además, constituye un objetivo necesario para su posterior mantenimiento y evolución en el tiempo. En el caso del software libre, el cumplimiento de este aspecto es fundamental para la creación de una comunidad de usuarios estable y duradera.

Se han escrito diversas guías que, en mayor o menor medida, aportan los conceptos necesarios para la creación y gestión de proyectos basados en software libre. En este apartado desarrollaremos esta cuestión mediante el artículo *Free Software Project Management HOW TO* de Benjamin Mako, que revisa las principales particularidades del proyecto desde el punto de vista práctico.

Web recomendada

B. Mako (2001). *Free Software Project Management HOW TO* (http://mako.cc/projects/howto).

Puesta en marcha

Antes de lanzar el proyecto basado en software libre, es muy importante diseñar una estructura sólida que permita soportar el posterior proceso de desarrollo con garantías suficientes.

En general, la estructura básica del proyecto debe dar respuesta a los siguientes aspectos:

- La necesidad de crear un proyecto nuevo, ya sea por ideas y objetivos propios o por la existencia de proyectos afines.
- La definición de las principales características de la aplicación (funcionalidades, licencia, numeración, etc.).
- La infraestructura básica de soporte a la difusión del nuevo proyecto y a la colaboración en su desarrollo (página web, correo de contacto, etc.).

Los desarrolladores

Una vez puesto en marcha el proyecto, nuestro segundo objetivo será la integración y consolidación de los usuarios y desarrolladores de la aplicación. Con relación a estos últimos, deberemos crear políticas y estrategias que nos permitan definir y estructurar su colaboración.

Las políticas de cooperación deben dar respuesta principalmente a dos objetivos concretos:

- la coordinación de la producción interna y externa, como la delegación de las responsabilidades y los protocolos de aceptación de las contribuciones.

- la gestión de la producción, como, por ejemplo, la estructura de ramas de desarrollo y los repositorios asociados.

Los usuarios

En los productos basados en software libre, los usuarios a menudo son también desarrolladores (y viceversa). Uno de los principales objetivos que debemos tener en cuenta son las pruebas o tests de la aplicación, ya sean funcionales, operativos, de calidad, etc.

La infraestructura de soporte

La actividad cotidiana del proyecto basado en software libre no se podría realizar sin una infraestructura de soporte adecuada a los objetivos cooperativos del mismo.

En la puesta en marcha del proyecto ya se han llevado a cabo las actuaciones básicas en este sentido, pero una vez está en funcionamiento, es necesario adecuar, mejorar y complementar los recursos existentes de acuerdo con la evolución del proyecto.

Recursos habituales

Algunos de los recursos habituales en los proyectos de software libre son: documentación, listas de correo, sistemas de control de errores y de versiones, foros, *chat*, *wiki*, etc.

La aplicación

Sin duda, el componente más relevante del proyecto y del que depende el resto de los aspectos considerados es la aplicación. Una de las características más relevantes que tiene que presentar la aplicación es que el usuario tenga suficientes garantías sobre el funcionamiento de cada una de las versiones que se lanzan al mercado.

El lanzamiento de versiones es un tema delicado que conviene meditar con calma. A grandes rasgos, tendremos en cuenta los siguientes aspectos:

- control de revisiones en términos de funcionalidades y corrección de errores (versiones alfa, beta, distribución candidata, etc.)

- cuándo lanzar la versión completa, es decir, cuándo estará preparado el código para ofrecer las garantías que esperamos y esperan los usuarios.

- cómo lanzar la versión (empaquetada, código fuente, formato binario, etc.).

Difusión del proyecto

Finalmente, y como ya hemos comentado al principio, difundir la existencia del proyecto es importante para el mismo, pero esta tarea se debe seguir considerando a lo largo del tiempo si queremos consolidar los fundamentos.

A medida que el proyecto avance, pensaremos en destacar el proyecto en listas de correo relacionadas con el software libre o en *Usenet*, incluir el proyecto en otros portales públicos (como *Freshmeat* o *Sourceforge*), o también anunciar las nuevas versiones de la aplicación en las listas de correo del mismo proyecto.

1.3. La gestión del proyecto

En este apartado profundizaremos en los aspectos de gestión del proyecto que, como promotores del mismo, deberemos tener en cuenta para dotarlo de garantías de éxito.

Los conceptos que presentamos en este apartado son complementarios a los expuestos en los apartados anteriores, puesto que posibilitan concretar y mejorar las diferentes actuaciones que se han considerado. En este sentido, es posible encontrar coincidencias directas o indirectas con estos argumentos.

Para relacionar los aspectos básicos de la gestión del proyecto basado en software libre, tendremos en cuenta las consideraciones consignadas en *Producing Open Source Software* de Karl Fogel, en especial el capítulo 5 titulado "Money".

Web recomendada

K. Fogel (2005). *Producing Open Source Sotware: How to Run a Successful Free Software Project* (capítulo 5 "Money"). (http://producingoss.com/en/money.html).

Financiación

Las particularidades de los proyectos de software libre hacen que muchas de las contribuciones estén subvencionadas informalmente (por ejemplo, cuando el trabajador de una empresa publica los ajustes que ha hecho en el código en el ejercicio de sus funciones).

Aun así, también se hacen donaciones y subvenciones que permiten obtener ingresos directos para el funcionamiento del proyecto, pero hay que tener en cuenta la gestión de estos fondos, ya que buena parte del apoyo que recibe un proyecto de software libre se basa en la credibilidad que merecen sus participantes.

Tipos de participación

Existen muchos tipos (y combinaciones posibles) de participación financiera en un proyecto de software libre. Además, en este modelo de financiación también influyen aspectos que no sólo dependen del mismo proyecto sino también de su entorno y contexto de actuación.

A grandes rasgos, la participación en un proyecto de software libre tiene relación con la colaboración de sus participantes, el modelo de negocio que explota la empresa que lo promueve (en caso de existir), las actuaciones de marketing llevadas a cabo, las licencias de los productos involucrados y las donaciones realizadas.

Contratos indefinidos

El equipo de desarrolladores de la aplicación es muy importante para el desarrollo del proyecto y su evolución futura. La estabilidad y permanencia de los participantes en sus puestos de responsabilidad permite solidificar las bases y la credibilidad del proyecto delante de la comunidad de usuarios.

Un proyecto estable

La credibilidad es fundamental para todos los actores relacionados directa o indirectamente con el proyecto, puesto que no se puede transferir a los eventuales sustitutos, y su pérdida puede afectar en mayor o menor medida al futuro de la aplicación y del proyecto, por lo que debemos tomar medidas adecuadas para controlar y gestionar activamente el proyecto.

Descentralización

Una de las características más relevantes (y deseables) de las comunidades de usuarios de software libre es la distribución y descentralización de las decisiones que se toman en el proyecto.

En este sentido, la organización del proyecto debe tener en cuenta esa estructura como forma de motivar y reforzar la comunidad de usuarios de la aplicación, de manera que el consenso surja de la misma interacción entre sus miembros.

Transparencia

El anterior aspecto de descentralización nos da una idea de la transparencia y justificación que debe reinar en la relación entre proyecto y comunidad.

Tanto los objetivos del proyecto como las líneas de evolución de la aplicación, deben estar claros y ser conocidos por todos los implicados en el mismo. En este sentido, la influencia del promotor sobre la evolución futura debe presentar un comportamiento honesto y transparente, de forma que garantice la credibilidad del proyecto[3].

(3)Por ejemplo, el manifiesto de Openbravo (http://www.openbravo.com/es/about-us/openbravo-manifesto/).

Credibilidad

La credibilidad del proyecto (tanto en conjunto como la de sus miembros) ha aparecido en muchos de los aspectos que hemos ido comentando hasta ahora. Su relevancia está muy relacionada con la comunidad de usuarios de software libre y supone una condición importante para el mantenimiento a lo largo del tiempo.

El dinero o la posición jerárquica no pueden generar la credibilidad necesaria en las actuaciones de cada uno de los miembros en cada momento. Es decir, la metodología, los procedimientos o protocolos establecidos, o el funcionamiento u operativa deben ser los mismos para todos sin excepción.

Contratos

La contratación de trabajadores es un aspecto a cuidar especialmente en los proyectos de software libre a causa de las repercusiones en la estructura y funcionamiento. Hay que velar para que todos los detalles y procesos relacionados con la contratación se mantengan abiertos y transparentes..

De hecho, es importante revisar y aceptar estos cambios con la colaboración de la comunidad de usuarios, hasta el punto de que en algunos casos puede ser preferible o deseable contratar directamente desarrolladores de la comunidad con permisos de escritura sobre el repositorio oficial (*committers*).

Recursos

El proyecto de software libre no sólo está basado en la evolución y mantenimiento del código de una aplicación basada en software libre, sino que debe considerar también otros aspectos de soporte complementarios.

Recursos complementarios

Este es el caso de la gestión de la calidad del código producido, la protección legal de las contribuciones, la documentación y utilidad de la aplicación y la provisión de recursos de infraestructura para la comunidad de software libre (páginas web, sistemas de control de versiones, etc.).

Estos recursos pueden motivar diferencias significativas en la difusión y popularización tanto de la aplicación como del proyecto en la comunidad de usuarios de software libre.

Marketing

Finalmente, aunque se trate de un proyecto basado en software libre, deben aplicarse medidas de marketing para su difusión y popularización tanto de la aplicación como del proyecto en general.

En este sentido es importante recordar que todo el funcionamiento del proyecto está expuesto al público en general, y cada una de las afirmaciones que se vierten pueden resultar fácilmente demostrables o revocables. El establecimiento de medidas para controlar la imagen y funcionamiento del proyecto han de permitir ganar credibilidad, transparencia y verificabilidad.

De entre estas medidas es importante remarcar la importancia de mantener una política abierta, honesta y objetiva respecto de los proyectos competidores. Por una parte porque sustenta un valor seguro para la comunidad de usuarios, y por la otra porque favorece el desarrollo de estrategias de coopetencia entre proyectos conexos.

2. La comunidad de usuarios

Tal como se ha puesto de relieve en el primer apartado de este módulo, el papel que juega la comunidad de usuarios de software libre es muy importante en el paradigma de desarrollo de software libre.

Tanto los usuarios como los desarrolladores que forman parte de la comunidad colaboran en el mantenimiento, soporte y evolución de la aplicación a lo largo del tiempo, asegurando así la cohesión y estabilidad del proyecto.

En consecuencia, su participación es fundamental para dotar de garantías los objetivos del proyecto, y debe ser considerada como tal por cualquier organización lucrativa que pretenda explotar una oportunidad de negocio basada en la producción de software libre.

En este sentido, la relación entre comunidad y empresa debe basarse en la credibilidad y la transparencia de todas las actuaciones y decisiones que se toman, de forma que ambas partes puedan extraer provecho de esta relación. No en vano el posicionamiento de la empresa con respecto a los productos basados en software libre debe estar bien definido y estructurado para favorecer la creación de una comunidad de colaboradores a su alrededor.

Hay que tener en cuenta que la comunidad de usuarios es una organización dinámica y evolutiva en el tiempo, por lo que será necesario establecer metodologías de gestión que permitan mantener la relación en su estado óptimo. Esta gestión incluye el establecimiento de procedimientos para identificar el estado actual de la comunidad, evaluar la calidad de las contribuciones al proyecto por parte de los miembros y definir los aspectos legales relacionados con estas contribuciones.

En los próximos apartados examinaremos cada uno de estos aspectos de forma individual.

2.1. Gestión de la comunidad

Para conseguir los objetivos el proyecto, la empresa que emprende un proyecto de desarrollo de software libre debe estructurar minuciosamente su relación con la comunidad de usuarios.

En el primer apartado de este módulo ya hemos mencionado los principales aspectos sobre los que fundamentar un proyecto de software libre. En caso de que una empresa actúe como promotora del proyecto, será necesario establecer y organizar una estrategia adecuada a los objetivos empresariales, pero teniendo en cuenta que tiene que ofrecer contrapartidas a la colaboración que espera obtener de la comunidad de usuarios.

En este sentido, y como en cualquier otro proyecto de software libre, aspectos como la credibilidad o la transparencia –entre otros– tendrán un papel muy importante en la creación de una comunidad de usuarios en torno al proyecto.

Ben Collins-Sussman y Brian Fitzpatrick han identificado y clasificado las diferentes estrategias *Open Source* que puede seguir una empresa basada en el desarrollo de software libre en la conferencia titulada "What's in it for me?" de OSCON 2007.

Web recomendada

B. Collins-Sussman; B. Fitzpatrick (2007). "What's in it for me?"
(http://www.youtube.com/watch?v=ZtYJoatnHb8).

Esta clasificación caracteriza los dos componentes principales de la relación entre empresa y comunidad:

- por una parte, la orientación, la estructura y el funcionamiento general del proyecto, así como la responsabilidad de la empresa en lo mismo.
- De la otra, los beneficios y los inconvenientes para la empresa y para la comunidad de usuarios resultantes de escoger una estrategia concreta para llevar a cabo el proyecto.

En este sentido, el trabajo de Collins-Sussman y Fitzpatrick describe una guía de buenas prácticas para formalizar una relación saludable entre empresa y comunidad de usuarios.

En los próximos apartados presentaremos de forma breve los principales rasgos de esta clasificación de estrategias *Open Source*.

Fake Open Source

Esta estrategia se basa en abrir o liberar el código fuente de la aplicación bajo una licencia no aprobada por OSI.

Web recomendada

Open Source Initiative (http://www.opensource.org/).

En realidad, no se trata de una estrategia *Open Source* real, puesto que no sólo se pierden los beneficios[4], sino que incluso algunos miembros de la comunidad podrían llegar a boicotear el proyecto.

(4)Por ejemplo, la mejora del software, la credibilidad del proyecto o la fortaleza de las relaciones entre empresa y usuarios.

A pesar de esto, el proyecto puede obtener cobertura mediática y atraer atención con un coste o esfuerzo relativamente bajo.

Throw code over the wall

Esta estrategia es similar a la anterior, pero esta vez la empresa abre o libera el código bajo una licencia aprobada por OSI, aunque sigue sin preocuparse o responsabilizarse por el futuro del proyecto.

Es decir, abriendo el código y olvidándose de él, la empresa difunde una imagen de poca credibilidad, puesto que libera una aplicación para la que no existe una comunidad de usuarios que permita mantener el proyecto activo. En este caso, es posible que se puedan crear comunidades alternativas que desarrollen el software al margen de los objetivos empresariales.

Develop internally, post externally

Esta estrategia se basa en desarrollar la aplicación de forma interna en la empresa, publicando los avances en un repositorio público.

De esta forma, la empresa mejora tanto las relaciones públicas con la comunidad de usuarios como la credibilidad dentro del mundo del software libre. Por su parte, la comunidad podría puntualmente colaborar en el proyecto. Aún así, el desarrollo totalmente interno incita a la creación de comunidades paralelas que no sigan el calendario empresarial (que genera una cierta desconfianza).

Open monarchy

Esta estrategia se basa en la exposición pública tanto de las discusiones como del repositorio de la aplicación, aunque los usuarios con derechos sobre él son internos a la empresa.

En este caso, se mejora sustancialmente la credibilidad y transparencia de las empresas y las aportaciones por parte de la comunidad (lo que redunda en mejor código), aunque la empresa sigue teniendo la última palabra en todas las decisiones que se toman. Este hecho constituye un riesgo para el mantenimiento de la comunidad a largo plazo, incluso existe riesgo de bifurcación (*fork*) en el proyecto.

Consensus-based development

Esta estrategia explota prácticamente todas las posibilidades de relación entre empresa y comunidad, puesto que prácticamente todo se realiza de forma pública.

En este caso, el proyecto se basa tanto en la toma de decisiones distribuida y descentralizada como en sistemas de funcionamiento meritocráticos entre los colaboradores.

Estas características redundan en un modelo sostenible a largo plazo con voluntarios de alta calidad, puesto que la empresa gana en credibilidad, transparencia y confianza delante de la comunidad y otras empresas basadas en software libre.

Aún así, los beneficios a corto plazo son escasos y el volumen de trabajo es significativo. En este caso, el rol de los líderes del proyecto es relevante para el funcionamiento estratégico de toda la organización.

2.2. Características de la comunidad

La comunidad de usuarios de software libre es una organización dinámica y evolutiva, en el sentido que existen diversos factores que influyen y moldean en mayor o menor medida su situación y tendencia futura.

Considerando un proyecto de software libre, es deseable crear cuanto antes una comunidad de usuarios sólida en torno a la aplicación, puesto que parte del éxito y de los objetivos del proyecto redundan en ella.

Una vez creada la comunidad, es importante programar actuaciones que permitan no sólo mantenerla estable sino también ampliarla y hacerla evolucionar al menos al mismo ritmo que lo hace el producto. Como paso previo a cualquier actuación en este sentido, es necesario conocer con precisión el estado actual de la comunidad de usuarios y la tendencia evolutiva que sigue en los últimos tiempos en relación con el proyecto.

Identificar con exactitud el estado actual de una comunidad de usuarios puede resultar relativamente complicado en la práctica, principalmente por sus características de distribución y descentralización[5].

(5)Esta problemática puede trasladarse y ejemplificarse con la problemática de evaluar la situación en un momento dado (fotografía instantánea) de un sistema distribuido o descentralizado.

Aún así, podemos tener en cuenta una serie de indicadores que nos permitan establecer una aproximación suficientemente realista como para tomar decisiones en este sentido.

En el artículo de Crowston y Howison "Assessing the Health of a FLOSS Community" se detalla una guía simple, pero efectiva, para identificar y evaluar el estado de una comunidad de usuarios de software libre. Esta guía considera los principales indicadores que se deben tener en cuenta para evaluar la salud de la comunidad y, por extensión, del proyecto basado en software libre.

Web recomendada

K. Crowston; J. Howison (2006). "Assessing the health of a FLOSS Community" (http://floss.syr.edu/publications/Crowston2006Assessing_the_health_of_open_source_communities.pdf)

En los próximos apartados presentaremos de forma breve algunas de sus conclusiones.

Ciclo de vida y motivaciones

Diversos autores están de acuerdo en considerar que los proyectos se inician en el seno de un grupo reducido de promotores para luego estructurarse y desarrollarse de forma pública.

Una vez puesto en marcha el proyecto, se debe iniciar una segunda fase que permita el refinamiento progresivo del concepto inicial. Es decir, la compartición de ideas, sugerencias y conocimientos ha de permitir revolucionar el concepto original. Este proceso no se puede realizar sin la cooperación de la comunidad de software libre.

Por otra parte, las motivaciones de los miembros de la comunidad para participar en el proyecto se relacionan principalmente con el desarrollo intelectual, el compartimiento del conocimiento, el interés por la aplicación, la ideología o filosofía subyacente al proyecto o al software libre, la reputación u obligación comunitaria.

Estructura y tamaño de la comunidad

La comunidad de usuarios de una aplicación basada en software libre puede estructurarse de muy diversas formas, teniendo en cuenta las actuaciones y decisiones de los promotores del proyecto y las características de la aplicación y/o de su producción.

En general, podemos considerar que la comunidad de usuarios de una aplicación es saludable si presenta una estructura jerárquica funcional adecuada a sus objetivos en torno a un núcleo activo de desarrolladores.

Estructura jerárquica

Este concepto puede relacionarse con la estructura de capas de una cebolla (*onion-shaped* en inglés), de forma que en el centro se sitúen los miembros más activos e implicados con el proyecto y en la capa más externa los menos participativos.

A grandes rasgos, podemos identificar las siguientes tipologías de miembros en un proyecto:

- Desarrolladores del núcleo de la aplicación, con derechos de escritura sobre el repositorio y un historial de contribuciones al proyecto significativo.

- Líderes del proyecto, que motiven y lleven el proyecto y su comunidad de usuarios a la madurez y estabilidad.

- Desarrolladores en general, que aportan código pero no tienen derechos de escritura sobre el repositorio. A menudo realizan tareas de revisión.

- Usuarios activos, que prueban la aplicación, informan sobre errores, crean documentación y enlazan el proyecto con los usuarios pasivos (entre otras actividades).

Nota

Esta primera clasificación de tipologías no es una estructura cerrada, puesto que cada proyecto la adapta a sus características particulares.

Procesos de desarrollo

El proceso de desarrollo de software libre puede resultar en muchas ocasiones poco formalizado en los proyectos, debido principalmente a la ausencia de mapas de ruta, asignaciones explícitas de trabajo o falta de priorización de las funcionalidades de la aplicación.

La organización del proyecto es una característica relevante para el funcionamiento y coordinación de la producción, aunque un cierto grado de duplicación de esfuerzos puede considerarse como un síntoma positivo de la relación e implicación de la comunidad en el proyecto.

Del mismo modo, el ciclo de evaluación y posterior aceptación de las contribuciones de los miembros de la comunidad al proyecto aporta información precisa sobre la salud de su funcionamiento. Por ejemplo, la refutación de una contribución puede demostrar una visión cohesionada y cualitativa del proyecto a largo plazo.

2.3. Gestión de la calidad

En ocasiones, la calidad del software libre ha sido foco de debate entre promotores y detractores, poniendo de relieve aspectos como la apertura del modelo de desarrollo o la capacitación de los colaboradores que contribuyen al proyecto, por ejemplo.

Como en cualquier proyecto de software, la producción de software libre debe establecer medidas de control de la calidad a lo largo de su ciclo de vida. Es decir, la calidad debe poder evaluarse y compararse con los baremos esperados en cualquier etapa de la producción y de la explotación, así como desde cualquier punto de vista (promotores, usuarios o comunidad).

En este sentido, si bien la apertura y descentralización del modelo de desarrollo del software libre favorece los mecanismos de control y gestión de la calidad, no constituye una solución por sí misma ni debe dejarse sin planificar en base a estas particularidades.

Para desarrollar los aspectos relacionados con la calidad en la producción de software libre, tendremos en cuenta el artículo "Managing Quality in Open Source Software" de Dhruv Mohindra, que realiza un estudio detenido sobre el control de calidad en entornos de software libre. En los próximos apartados revisaremos los principales conceptos que presenta el artículo.

Web recomendada

D. Mohindra (2008). "Managing Quality in Open Source Software"
(http://www1.webng.com/dhruv/material/managing_quality_in_oo.pdf).

La calidad en el software libre

En general, la calidad de una solución de software puede valorarse tanto por su arquitectura o diseño interno como por las funcionalidades que proporciona al usuario.

Las particularidades de apertura y descentralización del modelo de desarrollo del software libre constituyen una infraestructura que permite establecer políticas de gestión de la calidad basadas en la localización y resolución de problemáticas (entre otros aspectos). Aun así, en ocasiones la falta de claridad y/o estructura de los procesos de producción puede generar resultados que no son los esperados.

Evaluación de la calidad

Existen diversas metodologías y métricas formales para evaluar la calidad funcional de una aplicación. Las métricas cuantificables dependen en gran medida de la tipología del mismo software, por lo que deben ser elegidas en función de las características y objetivos de la aplicación.

En relación con la calidad no cuantificable, cabe destacar el papel que tiene la comunidad de software libre en este aspecto. Por una parte, los tests que realiza el equipo de calidad, y por otra, la propia actividad de los usuarios de la aplicación denotando las evidencias de errores de funcionamiento o de mejora del producto.

En este sentido, el mismo carácter y funcionamiento distribuido y descentralizado de la comunidad de usuarios también es importante para dotar de más garantías de calidad el proceso de producción.

Control y revisión

Un factor importante para la calidad del producto final es el control y la revisión de todo el proceso de desarrollo. En general, los proyectos de producción de software libre utilizan sistemas de control de versiones para soportar con eficiencia y eficacia la evolución de los diferentes componentes del proyecto.

Hay distintas formas de organizar el control y la revisión de la evolución del software, así como las ramas de desarrollo y repositorios (entre otros). En cualquier caso, conviene adaptar la metodología de producción y los sistemas de control y revisión de las evoluciones a las particularidades del proyecto y del producto que se está creando.

Mitos del software libre

A pesar del paso de los años, el software libre todavía arrastra algunos mitos –tanto positivos como negativos– que pueden influir en mayor o menor medida en su evaluación.

Estos mitos no tienen ninguna base sólida sobre la que fundamentar una gestión coherente y sostenible de la calidad, por lo que se debe concretar y evaluar cada uno de ellos por separado.

A continuación se comentan algunos mitos habituales relacionados con la calidad del software libre.

- El hecho de que el código fuente sea público no garantiza que sea seguro y/o de calidad, puesto que en cualquier caso depende del interés y revisión de la comunidad.

- La congelación de funcionalidades no aumenta la estabilidad de la aplicación por sí misma, porque lo importante es escribir buen código desde el principio.

- La mejor forma de entender un proyecto no es corregir sus eventuales fallos, puesto que la documentación es significativamente mejor para este objetivo.

- En general, los usuarios no disponen de la última versión del repositorio con la corrección de errores al día.

A grandes rasgos, los procesos de prueba y revisión, así como las discusiones públicas y la cultura *hacker* propias de la comunidad de usuarios, deben complementarse con una planificación y gestión activa de la calidad de la producción.

Esta gestión debe ir encaminada a cubrir eventuales lagunas en uno o más aspectos del producto, por ejemplo la planificación de la producción, desarrollo de las funcionalidades o la documentación de la aplicación.

Consideraciones adicionales sobre la calidad

En general, tanto la publicación del código fuente, como la incorporación de sistemas de gestión de errores y el hecho de compartir la responsabilidad sobre el producto entre todos los implicados son aspectos clave de la gestión de la calidad.

En este sentido, también resulta importante para la calidad general del proyecto considerar la transparencia en todas las actuaciones, confiar en el equipo de desarrollo, revisar y probar todas las partes del código fuente y promover tanto la filosofía de igual a igual como la importancia de hacerlo bien desde el principio.

2.4. Legalidad y contribuciones

En un proyecto basado en software libre con participación de la comunidad de usuarios, resulta especialmente relevante la gestión legal de las contribuciones de cada miembro implicado.

Esta gestión es importante tanto para los promotores del proyecto como para los miembros de la comunidad, puesto que establece las características de autoría y titularidad de los derechos del código resultante. Su relevancia también se ve especialmente influida por las implicaciones que puede tener la combinación de códigos de diferentes autores en un mismo producto.

Para desarrollar estos conceptos partiremos de la lectura del apartado 2.4 "Autores y titulares de derecho" de los materiales didácticos de la asignatura *Aspectos legales y de explotación del software libre*.

Web recomendada

M. Bain y otros (2007). *Aspectos legales y de explotación del software libre*. Universitat Oberta de Catalunya

(http://ocw.uoc.edu/informatica-tecnologia-i-multimedia/aspectes-legals-i-dexplotacio/materials/).

El autor

El autor de una obra es la persona física o jurídica que crea la obra, por lo que irrenunciablemente le pertenece la autoría de su creación original.

Las obras realizadas por diversas personas permiten distinguir entre varias situaciones:

- Una obra en colaboración es el resultado unitario de una composición de diversas partes que pueden ser explotadas independientemente.

- Una obra colectiva es la reunión de distintas contribuciones que no se pueden explotar de forma independiente.

- En una obra encargada (o con contrapartidas económicas), la autoría recae sobre la persona física o jurídica que realiza el encargo.

En el mundo del software libre, la autoría depende en gran medida de las consideraciones anteriores, teniendo en cuenta que en algunas ocasiones la transferencia de la titularidad puede resultar útil y práctica.

Por otra parte, las condiciones en las que se realizan las obras derivadas (preexistencia de contenido) pueden variar sustancialmente a causa tanto del autor como de la misma obra. En cualquier caso, las licencias libres deben especificar las condiciones de derivación y redistribución de las obras.

El titular original y el titular derivado

El titular original de la obra es siempre su autor. Aun así, algunos derechos sobre la obra pueden ser cedidos a otras personas, ya sean físicas o jurídicas.

En este caso, la persona que recibe la cesión de parte de los derechos sobre una obra se convierte en titular derivado de la misma. Conviene recalcar que sólo el titular de un derecho concreto puede conceder licencias sobre ese derecho.

Identificación del titular

Para poder ejercer los derechos anteriormente comentados tiene que ser posible identificar al autor de cada obra. Esto puede resultar complicado en el mundo del software libre, ya que los contribuyentes al proyecto pueden ser múltiples y variados.

Para paliar estos problemas, los proyectos basados en software libre mantienen listas de los autores que han contribuido. En algunas ocasiones, estos proyectos pueden requerir la cesión de la totalidad o parte de los derechos antes de aceptar la contribución.

3. Caso de estudio

En los apartados anteriores se examinan tanto el proyecto de software libre como la gestión de la comunidad de usuarios. Ambos apartados presentan los principales aspectos relacionados con la producción de software libre desde el punto de vista de gestión del proyecto.

Para finalizar este módulo, dedicaremos el último apartado a concretar buena parte de las ideas y propuestas presentadas anteriormente estudiando un caso concreto de empresa basada en software libre.

Los próximos apartados pretenden servir de guía para identificar y clarificar cómo una empresa basada en la producción de software libre implementa su metodología particular, formaliza y gestiona la relación con la comunidad de ususarios y cómo afronta muchas de las decisiones que debe tomar a medida que avanza el tiempo.

En este apartado nos centraremos en el caso de Openbravo, S. L.

3.1. La empresa

Openbravo, S. L. es una empresa dedicada a desarrollar soluciones profesionales basadas en software libre para las empresas.

Nota

Toda la información que se presenta en este apartado se ha obtenido principalmente de su sitio web (http://www.openbravo.com/).

Modelo de negocio

El modelo de negocio que explota la empresa se basa en la prestación de servicios relacionados con los productos que desarrolla. Tal y como se ha comentado en otros módulos, su estrategia de negocio está basada en el asociacionismo y la coopetencia entre empresas para explotar la misma oportunidad de negocio.

Estrategia empresarial

El modelo de negocio se implementa mediante *partners* que ofrecen servicios a los clientes finales (como por ejemplo la personalización y el soporte). En cierto modo, esta particular jerarquía entre productor, distribuidor (o *partner*) y cliente establece una atmósfera de cooperativismo con objetivos comunes.

Para completar esta estrategia, la empresa publica un manifiesto a modo de declaración de intenciones, que incluye tanto aspectos relacionados con el software libre (por ejemplo, la transparencia, la apertura o la colaboración), como los compromisos de la empresa respecto a terceras personas (por ejemplo, accesos libres o gestión de contribuciones).

Gestión y dirección

La gestión de la Empresa combina cargos internos y externos a la organización, tanto en el equipo directivo como en el Consejo de Administración, producto de la inversión externa que ha recibido la empresa y también de la particular metodología basada en software libre.

3.2. Los productos

Openbravo produce dos soluciones basadas en software libre que funcionan de forma independiente o en combinación. Ambos productos son distribuidos bajo licencias libres y con descarga directa a través de Internet.

Los productos que ofrece Openbravo son los siguientes:

- **Openbravo ERP**
  Openbravo ERP (*Enterprise Resource Planning*) es un sistema de gestión empresarial en entorno web, que integra de forma modular diversas funciones de gestión, como aprovisionamientos, almacén, producción o contabilidad.
  El producto está licenciado bajo MPL 1.1 y puede funcionar en diferentes entornos y sistemas de base de datos e integrarse con Openbravo POS.

  Entre las muchas informaciones que se proveen sobre el producto destaca el mapa de ruta de desarrollo del proyecto.

Web recomendada

Mozilla Public License 1.1 (http://www.mozilla.org/MPL/MPL-1.1.html).

Características principales de Openbravo ERP

http://sourceforge.net/projects/openbravo/.

- **Openbravo POS**
  Openbravo POS (*Point Of Sales*) es un sistema de terminal de punto de venta que puede ser integrado con Openbravo ERP.
  El producto está licenciado bajo GNU/GPL y puede funcionar en diferentes entornos y con diversos sistemas de base de datos. Está especialmente diseñado para terminales táctiles.

  Entre las informaciones que se proveen también destaca el mapa de ruta del producto de desarrollo del proyecto.

Web recomendada

GNU General Public License (http://www.gnu.org/licenses/gpl.html).

Características principales de Openbravo POS

http://sourceforge.net/projects/openbravopos/.

3.3. La comunidad de usuarios

La comunidad de usuarios de software libre juega un papel relevante en la estrategia empresarial de Openbravo. En los siguientes apartados mencionaremos los principales aspectos.

Estrategia *Open Source*

Para identificar la estrategia *Open Source* de Openbravo debemos tener en cuenta las particularidades de la metodología de desarrollo de los productos así como también la estructura de negocio que los explotan.

En este sentido, el núcleo de ambos productos es principalmente desarrollado de forma interna a la empresa, manteniendo repositorios públicos y una comunidad de usuarios activa a su alrededor. Por otra parte, en el desarrollo de los complementos, personalizaciones y extensiones sobre el producto original entran en juego tanto la comunidad de usuarios como los *partners*.

Este último caso debe analizarse por separado, puesto que corresponde a la explotación de una oportunidad por parte de una organización diferente.

Con todo esto, Openbravo presenta una estrategia *Open Source* que combina diferentes orientaciones:

- Para el desarrollo y revisión de los productos, la estrategia se aproxima a *Open Monarchy*, debido principalmente al desarrollo interno del núcleo de los productos, los repositorios públicos de código fuente, la aceptación final de los cambios sobre el núcleo a cargo de la empresa y la planificación del desarrollo de los productos (por ejemplo, los mapas de ruta establecidos).

- Para el desarrollo de los complementos (por ejemplo, la documentación, etc.), la estrategia se aproxima al *Consensus-based development*, debido principalmente a su desarrollo dentro de la comunidad de usuarios.

- Finalmente, la estrategia en el desarrollo de las extensiones y de las personalizaciones depende del desarrollador que las implemente. Si son proyectos realizados en el seno de la comunidad (mediante los recursos ofrecidos por Openbravo), posiblemente se aproximen al modelo de *Consensus-based development*, mientras que si son desarrolladas por los *partners* dependerán tanto de su estrategia particular como de las características del desarrollo.

Estrategia del *partner*

En el caso de que el *partner* desarrolle extensiones del producto original, la estrategia *Open Source* dependerá tanto de su filosofía empresarial como de las características del producto (por ejemplo, la licencia MPL es más flexible con los módulos propietarios que la GPL).

Estructura de la comunidad

La comunidad de usuarios de Openbravo ERP está definida y estructurada en forma de sistema meritocrático: existen varios niveles de colaboración y cada uno de ellos se define a partir de los conocimientos necesarios para este nivel, la cantidad de contribuciones, las responsabilidades y los privilegios.

En el caso de Openbravo ERP existen tres perfiles diferentes de colaboración (desarrolladores, expertos funcionales y probadores), mientras que en el caso de Openbravo POS sólo existen desarrolladores. Los miembros de la comunidad de usuarios se organizan y se distribuyen por proyectos activos en la comunidad.

Recursos a disposición de la comunidad

Openbravo dispone de diversos recursos (algunos de ellos en más de una lengua) tanto para la comunidad como para los *partners* o para los usuarios en general, entre los que destacan los siguientes:

Web recomendada

Desde el portal web de la empresa se puede acceder a todos los recursos mencionados (http://www.openbravo.com/).

- web corporativa
- área de *partners*
- wiki del proyecto
- portal de la comunidad de usuarios de Openbravo
- *blogs* de trabajadores
- forja de los productos (Openbravo ERP y Openbravo POS)
- *bug tracker*
- universidad
- listas de distribución de correo
- repositorio de código de Openbravo
- servicio de noticias de Openbravo

En general, la comunidad de usuarios tiene a su disposición una guía específica disponible en el wiki que describe cómo colaborar en el proyecto. También dispone de una lista exhaustiva de canales de comunicación a los que puede acceder. Adicionalmente, los mapas de ruta de cada producto completan la guía de recursos para la comunidad de usuarios.

3.4. Posicionamiento y evolución

La empresa nació en 2001 bajo el nombre de Tecnicia. En 2006 obtuvo financiación por más de seis millones de dólares cuando pasó a llamarse Openbravo. Ese mismo año liberó el código fuente de los productos que desarrolla bajo licencias libres.

En mayo del 2008, la ronda de financiación ascendió a más de doce millones de dólares, contando entre sus inversores a Sodena, GIMV, Adara o Amadeus Capital Partners.

A lo largo de los años, Openbravo ha recibido varios premios relacionados con el mundo de la empresa y del software libre, así como subvenciones del programa de fomento de la investigación técnica (PROFIT) del Ministerio de Industria, Turismo y Comercio de España.

Tanto la empresa como la comunidad de usuarios mantienen una tendencia evolutiva positiva, si tenemos en cuenta que el proyecto se sitúa en la actualidad entre los veinticinco más activos de SourceForge con más de un millón de descargas acumuladas a principios de 2009.

Web recomendada

Sobre los proyectos más activos de SourceForge:
http://sourceforge.net/top/mostactive.php?type=week.

Resumen

A lo largo del módulo se han presentado las principales características relacionadas con la creación, gestión y mantenimiento del proyecto de desarrollo de software libre, teniendo en cuenta la participación de la comunidad de usuarios.

En cierto modo, los fundamentos de la producción de software libre no difieren demasiado con respecto a las metodologías de desarrollo de software más tradicionales. Aun así, las características de la apertura del código y la presencia de la comunidad de usuarios moldean el funcionamiento y lo particularizan en muchos aspectos.

En lo que respecta al proyecto en sí, cabe destacar la importancia de identificar, definir y estructurar tanto los aspectos funcionales del proyecto (por ejemplo, la infraestructura, la gestión de versiones o las medidas de coordinación), como aquellos relacionados con el software libre (por ejemplo, la credibilidad, la transparencia o las tipologías de participación).

A estos aspectos se tienen que sumar aquellos factores relacionados con la comunidad de software libre, como la estrategia de gestión de la comunidad por parte de la empresa (estrategia *Open Source*), la metodología y ciclo de vida del producto, la gestión de la calidad y los aspectos legales de las contribuciones de los usuarios.

Finalmente, se ha presentado un caso de estudio representativo de buena parte de los aspectos examinados en los diferentes apartados.

Bibliografía

Bain, M. y otros (2007). *Aspectes legals i d'explotació del programari lliure*. Universitat Oberta de Catalunya <http://ocw.uoc.edu/informatica-tecnologia-i-multimedia/aspectes-legals-i-dexplotacio/Course_listing> [Fecha de consulta: febrero del 2009].

Collins-Sussman, B.; Fitzpatrick B. (2007). *What's in it for me? How your company can benefit from open sourcing code*. OSCON: 27 de julio del 2007 <http://www.youtube.com/watch?v=ZtYJoatnHb8> y diapositivas <http://www.red-bean.com/fitz/presentations/2007-07-27-OSCON-whats-in-it-for-me.pdf> [Fecha de consulta: febrero del 2009].

Crowston, K.; Howison, J. (mayo, 2006). *Assessing the Health of a FLOSS Community*. IT Systems perspectivas (pág. 113-115). <http://floss.syr.edu/publications/Crowston2006Assessing_the_health_of_open_source_communities.pdf> [Fecha de consulta: febrero del 2009].

Fogel, K. (2005). *Producing Open Source Software: How to Run a Successful Free Software Project.* <http://producingoss.com> [Fecha de consulta: febrero del 2009].

Mako, B. (2001). *Free Software Project Management HOW TO*. <http://mako.cc/projects/how-to> [Fecha de consulta: febrero del 2009].

Mohindra, D. (2008). *Managing Quality in Open Source Software*. <http://www1.webng.com/dhruv/material/managing_quality_in_oo.pdf> [Fecha de consulta: febrero del 2009].

Openbravo <http://www.openbravo.com/> [Fecha de consulta: febrero del 2009].

Raymod, E. (1997). *The cathedral and the bazaar* <http://www.catb.org/~esr/writings/cathedral-bazaar/> [Fecha de consulta: febrero del 2009].

Tawileh, A. y otros (agosto, 2006). *Managing Quality in the Free and Open Source Software Community* (págs. 4-6). Proceedings of the Twelfth Americas Conference on Information Systems. México: Acapulco. <http://www.tawileh.net/anas//files/downloads/papers/FOSS-QA.pdf?download> [Fecha de consulta: febrero del 2009].

Estrategias del software libre como negocio

Amadeu Albós Raya

PID_00145046

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

En el negocio del software libre, como en cualquier otro negocio basado en tecnología por lo general, intervienen diversidad de factores que influyen en mayor o menor medida en el éxito del proyecto. Buena parte de estos factores se tratan en el resto de módulos, como por ejemplo las características del mercado del software, los modelos de negocio o las particularidades de la producción de software libre.

En este sentido, el conjunto de actuaciones que permiten establecer una oportunidad de negocio válida y viable en la práctica debe ser minuciosamente puesto a punto para conseguir los objetivos que se pretende. Es decir, resulta fundamental trasladar las características del software libre como negocio al mercado objetivo real, con el fin de establecer una estrategia empresarial concreta y adecuada que permita aprovechar las ventajas que ofrece el software libre y controlar los inconvenientes que supone.

Esta estrategia debe reflejar la realidad del entorno y contexto empresarial, identificando y analizando las perspectivas de cada uno de los actores que intervienen en el mercado, para maximizar las garantías de éxito en la medida de lo posible.

A lo largo del presente módulo presentamos las principales particularidades que influyen en la estrategia de las empresas basadas en software libre, caracterizando los diferentes elementos en términos de ventajas o inconvenientes para el negocio.

Objetivos

Los objetivos que se deben alcanzar al finalizar este módulo son los siguientes:

1. Comprender la importancia de la estrategia en el negocio basado en software libre.

2. Identificar y valorar las ventajas de la explotación del software libre como negocio.

3. Identificar y valorar los inconvenientes ligados al negocio del software libre.

4. Relacionar y profundizar en las estrategias de los modelos de negocio del software libre.

1. La competitividad del software libre

En la actualidad, el software libre constituye una alternativa válida y viable al software propietario. Características como la modularidad de su desarrollo e instalación, la operativa basada en estándares y la constante evolución de las aplicaciones sustentan con suficiencia la competitividad del software libre.

A pesar de esta situación, esta competitividad puede no ser suficiente para el negocio del software libre si no se canalizan adecuadamente éstas y otras características. Es decir, para constituir un proyecto estable y confiable en el tiempo, es necesario definir una estrategia de negocio que aúne y coordine las ventajas que ofrece mientras que gestiona y controla sus inconvenientes.

En este primer apartado damos un breve repaso a las principales características que hacen del software libre una alternativa competitiva al software propietario.

Web recomendada

M. Boyer; J. Robert (2006). *The economics of Free and Open Source Software: Contributions to a Government Policy on Open Source Software* (cap. 3, "Advantages and disadvantages of FOSS").

<http://www.cirano.qc.ca/pdf/publication/2006RP-03.pdf>

Coste

En general, las aplicaciones basadas en software libre pueden obtenerse libremente y de forma gratuita a través de Internet. Esta filosofía de distribución se sitúa en las antípodas del modelo propietario, donde resulta habitual el pago por la explotación limitada del formato binario de la aplicación.

En consecuencia, el coste supone una ventaja competitiva importante para su adopción respecto de otras alternativas propietarias, puesto que pueden reducir significativamente la inversión necesaria para una implantación tecnológica (ya sea de nueva creación o una profunda actualización del sistema).

Por otra parte, la reducción del coste también puede resultar significativa en casos de evolución o especialización de una aplicación concreta, ya que, mientras el software libre garantiza la posibilidad de adecuación de la aplicación a los intereses particulares mediante el libre acceso al código fuente, el equivalente en software propietario puede requerir un desarrollo completamente nuevo.

Desarrollo, flexibilidad y modularidad

Si bien el desarrollo de una solución tecnológica basada en software libre puede ocasionalmente no diferir mucho respecto del equivalente propietario, la metodología basada en la colaboración y evolución conjunta entre empresa y comunidad de usuarios ofrece como ventaja la cooperación de escala.

Estas particularidades ofrecen un abanico de posibilidades que van desde el aprovechamiento de las economías de escala y la creación de mercados segmentados, hasta la flexibilidad y modularidad que permiten mejorar tanto la interoperabilidad e integración entre aplicaciones como su extensión y evolución. En definitiva, características que promueven la generación de oportunidades de negocio concretas.

Riesgo tecnológico

En términos generales, los riesgos ligados a la adopción de la tecnología afectan por igual tanto al software libre como al propietario, al menos desde un punto de vista tecnológico estricto.

En este sentido, y en el caso de aplicaciones o soluciones concretas, el riesgo tiene más relación con las capacidades y competencias específicas de cada una de ellas que con la tecnología o metodología utilizada para su desarrollo.

Seguridad, fiabilidad y ciclo de vida

Con el paso del tiempo, la evolución de las metodologías de desarrollo de software han permitido controlar más y mejor la calidad del software producido, especialmente en aspectos como adecuación y corrección de errores.

En este caso, la apertura del proceso de desarrollo del software libre y la colaboración de la comunidad de usuarios en dicho proceso le confiere una diferenciación sustancial respecto al modelo propietario. Es decir, resulta difícil que una empresa de producción de software propietario pueda equiparar los recursos humanos y temporales empleados en proyectos de software libre.

Esta particularidad del software libre favorece la competitividad y confiabilidad de las soluciones, tanto para las empresas como para sus clientes.

Soporte y documentación

En ocasiones, las aplicaciones basadas en software libre carecen del empaquetado habitual que acostumbra a ofrecerse en las aplicaciones equivalentes de software propietario. Desde el punto de vista comercial, esta situación consti-

tuye una fuente de oportunidades de negocio a diversas escalas, con la ventaja adicional que puede representar la especialización y la proximidad con el cliente.

Gestión del cambio

El software libre favorece la reestructuración de los valores integrados en el mercado tradicional: ofrece independencia, libertad, coste reducido y eficiencia de las inversiones; muchos de estos aspectos se presentan mitigados en el negocio tradicional del software.

Reestructuración de valores

El software libre ofrece independencia, libertad, coste reducido y eficiencia de las inversiones; muchos de estos aspectos se presentan mitigados en el negocio tradicional del software.

De forma adicional, también permite que las empresas puedan ajustar la estructura de costes y establecer estrategias de coopetencia entre proveedores afines o complementarios. Esta situación resulta más ventajosa, competitiva, menos arriesgada y más efectiva para sus participantes que los equivalentes consorcios del modelo propietario.

2. La perspectiva del cliente

Para el cliente de productos y servicios basados en software libre es muy importante identificar las ventajas e inconvenientes que le proporciona el modelo de software libre respecto del propietario, especialmente si éste se enmarca en un contexto de mercado tradicional fuertemente implantado.

Desde el punto de vista del cliente de productos de software, más que una diferenciación tecnológica en la arquitectura de los productos, las cuestiones económicas pueden resultar más relevantes para la implantación final de la tecnología. Estas particularidades deben tenerse en cuenta en la estrategia de las empresas si pretenden explotar el mercado con garantías de éxito.

En las siguientes secciones profundizamos en las ventajas e inconvenientes del negocio basado en software libre desde la perspectiva del cliente.

2.1. Ventajas

Las ventajas del software libre para el cliente constituyen una parte importante de la oportunidad de negocio para la empresa, ya que influyen en su posicionamiento en el mercado.

Efectos económicos

El software libre proporciona al cliente independencia de los proveedores tecnológicos, alternativas a productos y servicios propietarios (o incluso otras soluciones libres), explotación de una oferta creciente de software ligado a estándares y de las complementariedades consecuentes, así como el aprovechamiento de situaciones de software intercambiable (*commoditization*).

Costes

El aumento de la eficiencia y eficacia en la gestión de los costes tecnológicos puede resultar de mucha importancia para los clientes finales, ya sean particulares o empresas de cualquier tamaño.

El software libre favorece la introducción de cambios en la estructura de costes e inversiones tecnológicas del cliente mediante una gestión más eficiente y eficaz de las mismas.

Cambios en los costes

Se puede reducir el importe de la implantación aprovechando el software libre que se distribuye de forma gratuita o con la disminución de la actualización forzada de equipos en intervalos de tiempo poco espaciados. También puede utilizarse este ahorro para financiar servicios o inversiones tecnológicas a largo plazo (por ejemplo, el menor coste de mantenimiento de un sistema).

Web recomendada

J. García;; A. Romeo; C. Prieto (2003). *Análisis Financiero del Software Libre* (cap. 7) <http://www.lapastillaroja.net/capitulos_liberados_pdf/la_pastilla_roja_capitulo_7.pdf>

Por otra parte, el libre acceso al código fuente favorece la especialización y extensión de las aplicaciones basadas en software libre por parte del mismo cliente (o por alguna empresa especializada).

Valores éticos

En algunos casos, los valores éticos ligados al movimiento del software libre como la transparencia, la independencia, la igualdad o la cooperación pueden resultar adecuados a los fines y objetivos del cliente (o a la imagen que pretende difundir).

2.2. Inconvenientes

A pesar de los evidentes beneficios del software libre para el cliente, también presenta algunos inconvenientes que deben ser controlados y mitigados por aquellas empresas que pretendan explotar las oportunidades de negocio relacionadas.

Efectos económicos

El cliente puede tener reticencias para adoptar el software libre debido a los costes de cambio o a la compatibilidad con las soluciones que utiliza. En ocasiones, la valoración de alternativas puede sesgarse por la búsqueda de resultados o retornos a corto plazo, los mitos tecnológicos asociados al software libre o por la vinculación histórica con el software que explota actualmente.

Gestión del riesgo

Toda implantación de tecnología en una organización tiene un cierto riesgo asociado (incluso en un cliente particular), globalmente equiparables entre software libre y software propietario. Para el cliente, los matices que puedan existir entre ambas soluciones pueden resultar insalvables en determinadas condiciones, como por ejemplo cuando el historial del cliente presenta una o más tentativas fallidas de migración.

En ocasiones, el cliente puede no estar dispuesto a arriesgarse con novedades que puedan afectar al funcionamiento habitual de las áreas de procesos, tecnología y personal, obviando la necesidad de realizar ajustes en ellas para mejorar

la eficiencia en la organización tras una implantación tecnológica relevante. Este aspecto también puede resultar una fuente de posteriores problemas de funcionamiento y operativa sino se planifica detalladamente.

Gestión del coste

Parte de los costes de una implantación pueden resultar comunes tanto si se realiza con software libre como con software propietario. En ocasiones el cliente puede considerar que cambios de plataforma conllevan irremediablemente más costes ligados a la formación, al soporte, a la motivación del personal, o a la pérdida de productividad de la organización, por ejemplo. Puede resultar complicado contrarrestar estos argumentos, principalmente a causa de la dificultad para medirlos y cuantificarlos económicamente.

3. La estrategia empresarial

La visión del cliente y, por extensión, del mercado objetivo, resulta fundamental para definir una estrategia de negocio sólida para una empresa. Sin embargo, la empresa debe completar su estrategia teniendo en cuenta las ventajas e inconvenientes que le proporciona el modelo del software libre y, más concretamente, el modelo de negocio particular que explota.

La empresa basada en la explotación comercial del software libre debe ser consciente y realista del entorno en el que opera. Todas las particularidades del software libre, del cliente y del modelo de negocio que explota deben ser identificadas y analizadas para formalizar una estrategia realista y adecuada para la consecución de sus objetivos.

En este apartado nos centramos en un primer momento en las ventajas e inconvenientes del modelo del software libre para la empresa, para después analizar las estrategias ligadas a los modelos de negocio basados en software libre.

3.1. El modelo de software libre

Como en el caso del cliente, las particularidades del modelo del software libre influyen tanto en la definición del negocio como en las posibilidades de establecerse en el mercado y las capacidades de desarrollo empresarial en el largo plazo.

Ventajas

Las principales ventajas para el proveedor o empresa que explota de forma lucrativa el software libre se presentan a continuación.

- **Posicionamiento y diferenciación**

El software libre permite posicionar la empresa que lo explota en una situación favorable para la publicidad y el marketing positivo en el mercado, en el sentido de que la difusión puede resultar favorable a objetivos de afianzamiento, confianza, sostenibilidad y popularización de la empresa.

- **Mercado**

En el mercado de software tradicional, puede resultar difícil identificar y explotar nuevas oportunidades de negocio a causa de los efectos económicos de las políticas de negocio tradicionales. En este sentido, y como ya se ha co-

mentado en más de una ocasión, el software libre favorece la introducción de tecnologías novedosas (disruptivas) que permitan un sesgo diferencial aprovechable para nuevas oportunidades de negocio.

En consecuencia, el software libre favorece la penetración de nuevas empresas en el mercado tradicional rompiendo los efectos económicos que inmovilizan a los actores presentes en el mercado.

- **Desarrollo y distribución**

La libertad, facilidad y bajo coste de la distribución del software libre (normalmente, mediante descarga directa y gratuita por Internet), así como la cooperación, implicación y motivación de la comunidad de usuarios en su desarrollo, favorecen tanto la difusión como la adopción de las aplicaciones. Es decir, tanto la metodología de desarrollo como las particularidades en la distribución de las soluciones favorecen la eficiencia y eficacia del proyecto.

- **Costes y riesgos**

La carga y estructura de los costes y riesgos de las empresas basadas en software libre pueden resultar más ventajosos y competitivos que en modelos basados en software propietario a causa de la distribución y descentralización de parte de sus procesos entre los diferentes actores implicados.

- **Comoditización**

La comoditización del software es una situación globalmente ventajosa para el conjunto de actores, puesto que disminuye las barreras de entrada para nuevos productores de software, aumenta la competitividad del sector y, por tanto, se producen los mismos bienes con mayor eficiencia. Además de buscar la especialización y la diferenciación para explotar las oportunidades de negocio, también es posible actuar en un mercado totalmente comoditizado.

- **Innovación y creación de valor**

Las metodologías de producción y desarrollo abiertas y cooperativas redundan en una mayor eficiencia y eficacia, tanto del proceso creativo de innovación como de la creación y captura de valor por parte de la empresa. Es decir, con la apertura de los procesos productivos, la empresa deja de depender exclusivamente del personal interno para la innovación (que está sujeto a las limitaciones de tiempo y objetivos), y empieza a beneficiarse de las ideas y perspectivas de voluntarios, usuarios y clientes (cuya flexibilidad y motivación favorece el surgimiento de innovaciones interesantes).

Lectura recomendada

L. Morgan; P. Finnegan (2008). *Deciding on open innovation: an exploration of how firms create and capture value with open source software* (vol. 287, pág. 229-246). IFIP 2008.

De esta manera se cierra el bucle de retroalimentación entre empresa y clientes o usuarios (tratados como codesarrolladores), reduciendo el riesgo del proyecto y maximizando las garantías de éxito.

Inconvenientes

A continuación se tratan algunos de los problemas que se pueden presentar en las empresas basadas en software libre.

- **Efectos económicos**

Limitaciones

Por ejemplo, la cautividad de los clientes y la economía de las ideas impiden que la empresa pueda situarse en una posición preponderante en el mercado como podría pasar en algunos mercados copados por soluciones propietarias.

Algunos de los efectos económicos que propician la introducción de una nueva empresa en el mercado pueden también limitar la cantidad y calidad de sus operaciones.

- **Resultados**

Una de las consecuencias del punto anterior es que puede resultar difícil obtener grandes beneficios (al menos en la misma medida en que lo hacen las corporaciones de software propietario en la actualidad) o beneficios sostenibles durante un largo período de tiempo.

- **Comoditización**

La comoditización del software puede también tener un efecto negativo sobre las empresas basadas en el software libre si éstas no identifican y planifican bien la diferenciación de sus productos, servicios o incluso en las políticas de marketing. Es decir, una situación de bienes intercambiables puede repercutir en la composición y distribución del mercado si los productos no aportan una diferenciación sustancial a lo largo del tiempo.

Por otra parte, actuar sobre un mercado comotizado imposibilita obtener grandes márgenes de beneficios a causa de la relativa facilidad del cliente para cambiar de proveedor tecnológico. Es decir, la empresa debe ser realmente mejor o al menos tan buena como las competidoras del sector para mantener su posicionamiento, por ejemplo actuando sobre la velocidad de respuesta y la capacidad de adaptación.

- **Mitología**

A pesar del paso del tiempo, es posible que en algunos mercados aún persistan algunos mitos sobre el software libre que compliquen su implantación y despliegue. Combatir estos mitos puede resultar más o menos difícil en función de las mismas características del mercado, como por ejemplo el grado de implantación del software propietario o las eventuales tentativas fallidas de migración a software libre.

3.2. Producción de software libre

En general, si la aplicación desarrollada resulta exitosa entre los clientes potenciales, se pueden obtener ventajas relacionadas con la atracción de mejoras y complementos, la simpatía de la audiencia y de la comunidad, y menores costes de mantenimiento gracias a la participación de la comunidad.

Por el contrario, el desarrollo de software libre puede tener dificultades para recuperar la inversión inicial, que en ocasiones puede ser bastante significativa. Si bien es un problema común al mundo del software (ya sea libre o propietario), resulta más difícil vender copias de software libre que de otros modelos.

Modelos mixtos

La dualidad de los modelos mixtos (en general, versión pública y versión comercial) favorece la adopción y la difusión de la aplicación, pero conlleva algunos inconvenientes, como la poca participación de la comunidad en los objetivos empresariales o la necesidad de mantener un producto comercial interesante a lo largo del tiempo.

Este último aspecto puede implicar otros problemas si la gestión de la comunidad de usuarios por parte de la empresa no es la adecuada, como por ejemplo que la comunidad pueda desarrollar por su parte (y de forma pública) las extensiones propietarias de la versión comercial.

Software y servicios

En el caso de la prestación de servicios asociados a una aplicación libre, es posible llevar a cabo estrategias de coopetencia para ampliar el mercado objetivo, y para después segmentarlo mediante la diferenciación. En el caso de no poder establecer estrategias de coopetencia, el modelo presenta pocas barreras de entrada para competidores que, al disponer del código fuente, pueden dotarse de la infraestructura necesaria para competir como en un mercado tradicional.

Por otra parte, obtener ingresos cuantiosos únicamente a partir de servicios asociados puede presentar dificultades en mercados con fuerte presencia de innovadores y entusiastas de la tecnología.

3.3. Prestación de servicios ligados al software libre

La prestación de servicios presenta algunas ventajas respecto el equivalente en software propietario, como la ausencia de grandes costes derivados de las licencias, la calidad del producto y el acceso al código fuente. Estas características contribuyen a proporcionar servicios de forma eficiente y eficaz, lo que redunda en un valor añadido importante para el cliente.

Aun así, puede resultar difícil mantener los clientes en el largo plazo debido a la facilidad de entrar en el mercado y a la dificultad de diferenciar los servicios entre proveedores.

Pequeñas y medianas empresas

Las principales oportunidades de negocio se relacionan con el escaso empaquetado y distribución de las aplicaciones basadas en software libre (como la instalación, el soporte, la personalización o la formación), explotando nichos de mercado concretos.

Por el contrario, eventuales desarrollos a medida sobre una aplicación concreta pueden encontrar dificultades con la integración y compatibilidad con versiones posteriores. Del mismo modo, también puede resultar problemática la aparición de empresas competidoras en el mismo sector a causa del escaso margen para la coopetencia.

Grandes empresas

Para las grandes empresas, la participación en proyectos basados en software libre puede resultar relativamente fácil debido a la existencia de infraestructura y organización previa. El software libre permite además ahorrar costes, mejorar la imagen de marca en aspectos como la fiabilidad, la solidez, la confianza, la estabilidad o el soporte profesional.

Aun así, formalizar una imagen de marca no es fácil a corto plazo. La predominancia de grandes corporaciones de software propietario en el mercado complica el posicionamiento, y el riesgo de proyectos de gran envergadura también es mayor.

3.4. Mercados auxiliares

En general, los modelos de negocio asociados a mercados auxiliares pueden servir como complemento de estrategias principales. Sin embargo, pueden resultar adecuados y viables como estrategia principal en mercados con poca competencia o con requisitos de diferenciación o especialización.

Hardware

El mercado auxiliar de hardware puede resultar válido para explotar mercados que requieran de especialización en los productos, como por ejemplo los servicios integrados, de alto rendimiento o con menor coste de compra para el cliente. Es decir, mercados donde los sistemas propietarios pueden no tener interés y el software libre puede suponer una diferenciación significativa para el cliente.

Los principales inconvenientes provienen de la capacidad para asumir los costes de producción y desarrollo si el mercado objetivo es limitado o existe una fuerte competencia de precios. En algunas ocasiones, la dificultad para recuperar la inversión inicial puede hacer que resulte poco adecuado para pequeñas y medianas empresas.

Otros mercados

Los mercados auxiliares como la venta de libros o el *merchandising* pueden resultar competitivamente equiparables a sus homólogos basados en software propietario, teniendo en cuenta las particularidades ligadas al software libre, como por ejemplo las complementariedades respecto el producto original o la difusión de valores éticos.

Resumen

El modelo de software libre constituye una alternativa válida y viable al software propietario, formalizando características competitivas para su implantación con objetivos de muy diversa índole como el coste y la flexibilidad.

Estas características presentan ventajas e inconvenientes para los principales actores implicados en el mercado del software. En ocasiones, aquellos aspectos que presentan una ventaja para unos suponen un inconveniente para otros, poniendo de relieve la necesidad de formalizar una estrategia de negocio realista que permita garantizar los objetivos con eficiencia y eficacia.

Para desarrollar la estrategia, la empresa basada en software libre debe tener en cuenta las implicaciones del modelo del software libre tanto en el cliente como en su propio funcionamiento:

- Para el cliente, el software libre le permite combatir los efectos económicos de un mercado tradicional y gestionar mejor el coste de la implantación, a costa de asumir un cierto riesgo.

- Para la empresa supone una oportunidad de negocio basada en la diferenciación y en la gestión eficiente de costes y riesgos, a costa de limitar su posicionamiento en el mercado y los resultados que puede obtener.

La formalización de la estrategia empresarial permite explotar más y mejor las ventajas del software libre en el contexto de actuación de la empresa, a la vez que gestiona y mitiga los inconvenientes que pueden limitar sus garantías de éxito.

Bibliografía

Boyer, M.; Robert, J. (2006). *The economics of Free and Open Source Software: Contributions to a Government Policy on Open Source Software*. Centre Interuniversitaire de recherche en analyse des organisations (CIRANO), 2006RP-03 <http://www.cirano.qc.ca/pdf/publication/2006RP-03.pdf> [Consulta: marzo 2009].

García, J.; Romeo, A.; Prieto, C. (2003). *Análisis Financiero del Software Libre*. La Pastilla Roja, capítulo 7.

<http://www.lapastillaroja.net/capitulos_liberados_pdf/la_pastilla_roja_capitulo_7.pdf> [Consulta: marzo 2009]

Ghosh, R. A. UNU-MERIT, N. L. (2006). *Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies sector in the EU*

<http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf> [Consulta: marzo 2009].

Iansiti, M.; Richards, G. L. (2006). *The Business of Free Software: Enterprise Incentives, Investment, and Motivation in the Open Source Community.*

<http://www.hbs.edu/research/pdf/07-028.pdf> [Consulta: marzo 2009].

Morgan, L.; Finnegan, P. (2008). "Deciding on open innovation: an exploration of how firms create and capture value with open source software". En: G. León; A. Bernardos; J. Casar; K. Kautz; J. de Gross (eds). "International Federation for Information Processing". *Open IT-Based Innovation: Moving Towards Cooperative IT Transfer and Knowledge Diffusion* (vol. 287, págs. 229-246). Boston: Springer.

West, J.; Gallagher, S. (2006). "Patterns of Open Innovation in Open Source Software". En: Henry Chesbrough; Wim Vanhaverbeke; Joel West (eds.). *Open Innovation: Researching a New Paradigm* (pág. 82-106). Oxford: Oxford University Press.

<http://www.openinnovation.net/Book/NewParadigm/Chapters/index.html> [Consulta: junio 2008].

Wheeler, D. (2007). *Why Open Source Software?*

<http://www.dwheeler.com/oss_fs_why.html> [Consulta: marzo 2009]

El software libre, ¿un nuevo modelo económico?

Amadeu Albós Raya

PID_00145045

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

En el presente módulo examinaremos el paradigma del software libre desde el punto de vista de modelo económico. Es decir, estudiaremos el ajuste y la viabilidad del software libre como modelo de funcionamiento económico sostenible a largo plazo.

En el estudio del software libre como modelo económico estaremos limitados por la relativa juventud del negocio basado en software libre. Aun así, considerando que las reglas económicas del mercado por lo general no han sido modificadas, basaremos el estudio en la diferenciación que supone el negocio del software libre con respecto a los mercados tradicionales. Este punto de vista nos permitirá obtener una primera aproximación realista sobre las cualidades del software libre como modelo económico.

En un primer momento revisaremos las bases que sustentan el paradigma del software libre y, por ende, su funcionamiento y posibilidades. Es decir, veremos aquellas características conceptuales ligadas a la filosofía subyacente y operativa del modelo, como por ejemplo la producción social.

A continuación analizaremos las consecuencias del modelo basado en software libre desde diferentes puntos de vista, teniendo en cuenta las diferencias que supone con respecto a los modelos tradicionales de producción de software y modelos de negocio. La proyección de estos conceptos nos debe ayudar a comprender mejor cuál puede ser el encaje del modelo del software libre en el mercado en un futuro próximo.

Por último estudiamos cómo se relaciona el modelo del software libre con la validez y la viabilidad de las empresas que se basan en él, denotando la importancia de conjugar tanto estrategia como oportunidad.

Objetivos

Los objetivos que se deben alcanzar al finalizar este módulo son los siguientes:

1. Familiarizarse con los aspectos económicos del modelo ligado al software libre.

2. Entender los fundamentos y las implicaciones del modelo de software libre con respecto al modelo tradicional.

3. Comprender la diferenciación que supone el modelo de software libre y evaluar su adecuación a la creación de valor para el mercado.

4. Profundizar en la validez y viabilidad del modelo de software libre y en los modelos de negocio explotables.

1. Las bases del modelo

Del software libre conocemos muchas de las características tecnológicas que, en mayor o menor medida, pueden resultar similares a las de cualquier software propietario. Es decir, las diferencias fundamentales (de haberlas) entre el software libre y el propietario no están basadas en los aspectos internos o externos del producto.

A grandes rasgos, la tecnología aplicada a un producto (por ejemplo, el diseño, la arquitectura o la implementación particular), no justifican por sí solas una diferenciación substancial entre modelos libres y propietarios, al menos desde el punto de vista estricto del producto acabado.

> Las principales diferencias entre el software libre y otros paradigmas de producción de software (especialmente el propietario) se centran en las particularidades del modelo de desarrollo, la comunidad de usuarios y en la diferenciación del valor añadido del producto.

Estas diferencias no se basan en aspectos tecnológicos propios de la aplicación o del software para desarrollarlo, sino en las características e implicaciones subyacentes a su producción. Es decir, condensa una orientación particular para crear valor en productos y servicios que difieren del punto de vista tradicional.

Como se explica en los módulos precedentes, a lo largo de los últimos años se han perfeccionado modelos de negocio que explotan estas características diferenciales en un mercado tradicional. En cualquier caso, el valor principal no recae en el software en sí, sino en el capital que se adquiere cuando se adopta.

Este capital formaliza los fundamentos del software libre. Es decir, el software libre se fundamenta en la producción social y la cultura en red que no sólo lo hacen posible sino que también potencian sus capacidades y efectos.

En los siguientes apartados desarrollaremos ambos conceptos de forma breve. En un primer momento examinaremos las principales características de la producción social y después caracterizaremos la cultura en red y sus efectos en la economía que sustenta el software libre.

1.1. La producción social

Posiblemente, tanto los avances en las comunicaciones globales como la democratización de la tecnología a lo largo de las últimas décadas han influido de diferentes formas en lo que hoy en día consideramos como software libre.

Es decir, la facilidad de acceso a la información y la voluntad de cooperación no son características únicas del software libre, sino que estructuran una base para el desarrollo de alternativas válidas y viables en multitud de campos.

Si bien en la actualidad existen múltiples iniciativas ligadas en mayor o menor medida a la producción social, las organizaciones empresariales encuentran en este modelo una manera de incentivar la creación y captura de valor para sus modelos de negocio.

Una muestra de producción social

Por ejemplo, la Wikipedia (http://www.wikipedia.org/).

Yochai Benkler, en su libro *The Wealth of Networks*, estudia en profundidad esta cuestión. A continuación revisaremos algunos de los aspectos más relevantes que caracterizan la producción social.

Lectura recomendada

L. Morgan; P. Finnegan (2008). *Deciding on open innovation: an exploration of how firms create and capture value with open source software* (vol. 287, pág. 229-246). IFIP.

La economía de la información

La información es un bien público que tiene implicaciones económicas en diferentes niveles gracias a las tecnologías de la información.

Web recomendada

Y. Benkler (2006). *The Wealth of Networks: How social production transforms markets and freedom.* (http://www.benkler.org/Benkler_Wealth_Of_Networks.pdf).

La innovación, como creación de nueva información, puede verse negativamente afectada por situaciones de restricción o control, mientras que puede ser facilitada por la apertura y colaboración en la producción de la información, el conocimiento y la cultura.

En este sentido, la producción o innovación en redes colaborativas o de igual a igual genera una espiral de oportunidades caracterizada por la motivación y la eficiencia con el soporte de la tecnología.

Redes colaborativas

En inglés, *Peer-to-Peer*. En este caso, el término hace referencia al funcionamiento de la comunidad y no al soporte arquitectural o tecnológico que sustenta la comunicación.

El desarrollo y distribución de la información

El desarrollo y distribución de la información puede seguir diversidad de patrones específicos en función de la distribución de libertad entre productores y consumidores. En general, cuanta más libertad se otorga al productor, menos libertad obtiene el consumidor.

Los canales de distribución de la información influyen en la forma de compartirla. El sentido direccional de la transmisión y los objetivos de la misma influyen también en cómo se comparte la información.

En cualquier caso, las licencias y las patentes pueden restringir el flujo de información, mientras que el crecimiento cuantitativo de la red no tiene por qué fragmentarla o limitarla.

Las implicaciones de la producción social

Benkler afirma que nuestra forma de percibir la estructura de funcionamiento del mundo que nos rodea está en plena transformación, especialmente en cuanto a la forma de colaborar e interactuar entre todos en la integración de ideas y conocimientos para crear nuevo conocimiento.

1.2. Economía y cultura en red

Las implicaciones de la producción social se han hecho patentes en multitud de campos en los últimos tiempos, especialmente en el software libre. La interacción de conocimientos y el refinamiento de ideas es, en la actualidad, una buena forma de motivar y profundizar en el desarrollo de un concepto.

Esta visión de la producción como colaboración para alcanzar cualitativamente un objetivo concreto contrasta con la visión más tradicional del mercado de ideas y conocimientos, donde la importancia recae más en la adopción final del producto que en el consenso, la adecuación o la calidad.

En la publicación de David Bollier *When Push Comes to Pull: The New Economy and Culture of Networking Technology* se examina con detalle cómo la evolución de la tecnología de la información ha permitido crear un nuevo punto de vista que contrasta con la centralización y jerarquía del modelo tradicional.

Web recomendada

D. Bollier (2006). *When Push Comes to Pull: The New Economy and Culture of Networking Technology.* (http://www.aspeninstitute.org/atf/cf/%7bDEB6F227-659B-4EC8-8F84-8DF23CA704F5%7d/2005InfoTechText.pdf).

En los siguientes apartados examinaremos brevemente las principales características económicas y culturales de la cultura en red que considera Bollier.

El modelo *push* y el modelo *pull*

El modelo *push* se basa en la producción en masa, anticipando la demanda de los consumidores y gestionando de forma dinámica el tiempo y la ubicación de los recursos de producción.

El modelo *pull* se basa en la apertura y la flexibilidad de las plataformas de producción que se utilizan como recursos. Este modelo no anticipa la demanda de los consumidores, sino que personaliza los productos en función de la demanda mediante procesos rápidos y dinámicos.

Redes de creación de valor

En los modelos *pull*, el hecho de compartir tanto la información como las buenas prácticas mejora sustancialmente el corpus de conocimientos de todos los miembros de la red.

Esta red fomenta y cohesiona modelos de negocio abiertos basados en la creación de valor y personalización o diferenciación de los productos.

En este sentido, las plataformas del modelo *pull* formalizan, mejoran y flexibilizan la innovación y evolución a través de la comunidad, sin tener que asumir los costes que supondría su implementación similar en un modelo *push*.

El mercado objetivo

Los modelos *push* tienen éxito en las áreas donde los consumidores no tienen muy claro lo que quieren y prefieren realizar la selección sobre tipologías predefinidas.

En cambio, en los modelos *pull*, los consumidores quieren formar parte del proceso de elaboración y selección, en el sentido de que quizás no saben qué es lo que quieren, pero están seguros de querer participar y formar parte del proceso.

La producción

Los modelos *push* tienden a buscar alternativas de producción que puedan resultar más competitivas económicamente (por ejemplo, menor coste de producción), mientras que los modelos *pull* tienden a buscar sobre todo las mejores formas de aportar valor a la red de producción.

Esta particular orientación de los modelos *pull* favorece la escalabilidad de la red de producción, así como el reencuentro de los mejores participantes para la especialización de la producción.

La cooperación

En los modelos *pull* se favorece la creación de relaciones basadas en la confianza, el hecho de compartir conocimientos y la cooperación entre los miembros que forman parte de la red para el beneficio de todos.

En muchas ocasiones, esta particular filosofía se transforma en un régimen de gobierno colectivo para gestionar de forma equitativa y sostenible los recursos compartidos.

En este sentido, las empresas basadas en los modelos *pull* deben ofrecer garantías para el reconocimiento de los miembros de la red, puesto que el modelo se basa en la confianza y en la creación de valor.

La educación

Los modelos *push* permiten centrar la actividad de los estudiantes en construir conocimiento estático, a modo de entrenamiento para la posterior sociedad jerarquizada.

Los modelos *pull* promueven formas de educación alternativas, en el sentido de que las tecnologías de la información hacen posible que los estudiantes entren en un flujo de actividad dinámica y puedan acceder a multitud de recursos independientes para crear su propio corpus de conocimiento (y de compartirlo a su vez).

2. Las características del modelo del software libre

Los fundamentos en los que se basa el software libre formalizan una estructura donde la cooperación y el hecho de compartir conocimientos entre sus miembros permite innovar, producir y hacer evolucionar el conocimiento global.

Sin lugar a dudas, la creación de valor es un objetivo importante para todos los miembros de la comunidad (ya sean usuarios, desarrolladores, etc.) y para el modelo en sí. En este sentido, la descentralización, la libertad y la independencia que rigen la comunidad ofrecen garantías para consolidar y cohesionar tanto la producción como el capital social.

El modelo del software libre se fundamenta en la diferenciación de los valores que rigen el mercado tradicional, tanto desde el punto de vista de desarrollo de software como de la apreciación del valor creado.

Si bien es cierto que desde un punto de vista clásico algunas de las características del modelo del software libre son aplicables también a otros paradigmas de desarrollo y de creación de valor, el modelo del software libre introduce novedades en la percepción y apreciación de los valores ligados al mercado tradicional.

En este apartado del módulo examinaremos cuáles son las características del modelo del software libre en comparación con las de un modelo tradicional, con el objetivo de valorar la diferenciación real que propone el modelo en la práctica diaria.

En un primer momento examinaremos el modelo desde el punto de vista del desarrollo del software, para luego analizar las implicaciones de la diferenciación como paradigma basado en la producción social.

2.1. El desarrollo de software

La metodología de desarrollo de software libre es posiblemente uno de los factores que popularmente se consideran diferenciadores con respecto a otros paradigmas de desarrollo de software, por ejemplo el modelo propietario. ¿Es realmente así?

Desde el punto de vista de producción de software, el desarrollo de software libre tiene importantes puntos de coincidencia con otros modelos de desarrollo, por ejemplo el propietario, dado que las metodologías de producción tienen una cierta independencia con respecto a las implementaciones particulares.

Pero que la producción de software pueda resultar más o menos coincidente con respecto a otros modelos o que algunos de los requisitos de libertad sobre el código sean más o menos necesarios en la práctica, no implica que no pueda existir una diferencia significativa en otros aspectos que permita valorar el conjunto como novedoso.

En este sentido, el artículo de Fuggetta titulado *Software libre y de código abierto: ¿un nuevo modelo para el desarrollo de software?* analiza en profundidad este y otros aspectos de las diferencias entre el modelo de desarrollo del software libre y el modelo de desarrollo del software propietario. En los siguientes apartados se revisan de forma breve algunas de sus conclusiones.

Web recomendada

A. Fuggetta (2004). *Software libre y de código abierto: ¿un nuevo modelo para el desarrollo de software?* (http://alarcos.inf-cr.uclm.es/doc/ig1/doc/temas/4/IG1-t4slibreabierto.pdf)

El contexto

El éxito del software libre se puede atribuir a una variedad de aspectos tecnológicos y económicos ligados a la innovación y a la producción del mismo.

Las características de descentralización, cooperación y libertad de uso y explotación convierten al software libre en el caballo de batalla de una nueva filosofía para abordar y solucionar problemáticas de diversa índole.

Según Fuggeta, muchas de las creencias sobre el software libre pueden ser aplicadas también al software propietario, por lo que es conveniente realizar un análisis exhaustivo de la temática.

El proceso de desarrollo

Desde el punto de vista tecnológico, el desarrollo del software libre no supone un nuevo paradigma, puesto que la mayoría de proyectos se sustentan en un número limitado de colaboradores, mientras que las metodologías de desarrollo incrementales y evolutivas no son exclusivas del software libre.

En cambio, el software libre ha conseguido motivar tanto a desarrolladores como a usuarios a involucrarse en el proyecto, compartiendo y conectando el desarrollo y la evolución del software con las necesidades de la comunidad.

La protección de los derechos de los clientes

Los problemas relacionados con la protección de los clientes aparecen principalmente cuando se trata de un paquete de software, puesto que en los desarrollos a medida el cliente ya es propietario del código.

En el caso de paquetes de software podría resultar suficiente tener acceso al código fuente sin poder modificarlo y redistribuirlo a su vez. Por otro lado, el soporte al usuario por parte de la empresa debería regirse por normas que faciliten la entrega del código en el caso de que la empresa no pueda asumir su mantenimiento.

La difusión del conocimiento

La propagación del conocimiento mediante el acceso al código fuente es insuficiente, puesto que las materias relacionadas con la ingeniería del software demuestran que es necesario disponer de documentos que describan la arquitectura del software.

Por otra parte, en el caso de que se pudiera propagar este conocimiento, sólo sería necesaria la publicación de su código fuente (sin derecho a copiar y redistribuir el software).

El coste

El hecho de que el software esté publicado bajo una licencia libre no significa que no pueda ser comercializado o que su desarrollo no tenga un coste asociado (aunque no sepamos qué magnitud tiene).

Por otra parte, que no se pueda cuantificar o centralizar su coste no significa que este no sea asumido de forma distribuida por los colaboradores, incluso de forma indirecta por empresas que poco o nada pueden relacionarse con el mundo del software.

La efectividad del modelo de negocio

Las principales modelos de negocio que explotan realmente el software libre son los que se dedican al desarrollo y distribución de paquetes de código abierto puros o a plataformas de software libre y propietarias. Otras formas de negocio pueden aplicarse en mayor o menor medida tanto al software libre como al propietario.

Por otra parte, por el momento nada indica que una empresa basada únicamente en servicios pueda ser rentable en el tiempo.

Sobre la rentabilidad

En el libro *The Business of Software*, Michael Cusumano argumenta que las compañías de software dependerán cada vez más de la combinación de ingresos entre licencias y servicios.

La industria del software

Europa no dispone de una estrategia industrial que permita cohesionar las actuaciones de las diferentes empresas implicadas. En este sentido, apoyar el software libre no es una estrategia en comparación con la creación de productos innovadores.

2.2. El paradigma cooperativo

Si bien algunas de las características del modelo del software libre no son innovadoras desde una perspectiva clásica, tal como hemos examinado anteriormente, sí lo son aquellas que motivan precisamente un cambio en la perspectiva del mercado.

Para valorar en profundidad la diferenciación que supone el modelo de software libre con respecto a otros modelos tradicionales hay que valorar los aspectos de producción y creación de valor y de conocimiento en los que se basa el modelo.

En el artículo *Open Source Paradigm Shift*, Tim O'Reilly desgrana estas y otras características del software libre como diferenciadoras y creadoras de una ventaja competitiva explotable de forma lucrativa. En los próximos apartados realizaremos un breve repaso a algunas de sus conclusiones.

Web recomendada

T. O'Reilly (2004). *Open Source Paradigm Shift*.
(http://www.oreillynet.com/pub/a/oreilly/tim/articles/paradigmshift_0504.html).

El cambio

El software libre supone un cambio profundo en la estructura del mercado de referencia que, en muchas ocasiones, tiene implicaciones que van más allá de las consideradas por sus creadores.

Los cambios se sustentan en la calidad del producto, la disminución del coste de producción y la explotación de estándares, así como en la diferenciación en las áreas de marketing, distribución y logística.

El software como *commodity*

Considerando un contexto de comunicaciones permanentes estandarizadas como el actual, todas las aplicaciones de comunicación son intercambiables entre sí (por ejemplo, un navegador web). Es decir, la explotación de estándares provoca que el software pueda ser considerado como *commodity*.

En este sentido, cuando el potencial de generar beneficios de una aplicación se diluye a causa del proceso de *comoditization*, aparece un nuevo mercado para productos propietarios, especialmente si éstos explotan la red global de comunicaciones.

Por otra parte, el software libre sigue siendo un modelo viable para empresas prestadoras de servicios, aunque no se puede esperar un margen de beneficios similar al de las grandes compañías de software actuales.

Colaboración en red

La cultura del intercambio de software ha crecido desde sus inicios al mismo ritmo que Internet, cuya arquitectura participativa se implementa en prácticamente todas sus funcionalidades.

El software libre constituye el lenguaje natural de la comunidad en red, dando lugar a un estilo de colaboración y participación particular entre sus miembros. Esta colaboración es fundamental para el éxito y la diferenciación de las aplicaciones líderes en Internet, dado que ha puesto de relieve la importancia de tratar a los usuarios como co-desarrolladores del software.

Personalización y *software-as-service*

En la actualidad, estamos acostumbrados a considerar una aplicación como un artefacto estático y no como un proceso. Los programas requieren ingeniería para ser creados, pero los lenguajes dinámicos que permiten la cohesión de los componentes (por ejemplo, los *scripts* de gestión de datos), ofrecen una perspectiva de proceso dinámico y evolutivo de la aplicación.

Buena parte de los servicios que se ofrecen por Internet (por ejemplo, un buscador) requieren revisiones y actualizaciones constantes para realizar su función correctamente. Esta situación genera un nuevo paradigma de negocio relacionado con el mundo de los ordenadores y de la tecnología de la información, en general, especialmente la explotación del software como servicio.

El sistema operativo Internet

Podemos tratar Internet como un único ordenador virtual que construye un sistema operativo a partir de la unión de diferentes piezas pequeñas y permite la participación de cualquiera en la creación de valor.

Los valores de la comunidad de usuarios del software libre son importantes para el paradigma, puesto que promueven un espíritu de buscar y compartir el conocimiento.

El proceso de *commoditization* de la tecnología es parte del proceso que permite que la industria avance para crear más valor para todos. Para la industria es esencial encontrar el equilibrio que permita crear más valor que el que se captura con participantes individuales.

3. La validez y la viabilidad del modelo del software libre

En los anteriores apartados hemos examinado tanto los fundamentos sobre los que se sostiene el modelo del software libre como las características que lo diferencian de los modelos más tradicionales.

Para valorar la sostenibilidad del modelo del software libre a largo plazo necesitamos muchos más datos que los que se poseen en la actualidad, es decir un corte temporal mucho más amplio que permita una equiparación mucho más precisa con los modelos tradicionales.

> El paso del tiempo determinará si el software libre constituye un nuevo modelo económico y cuáles serán las características y condiciones que lo permitirán.

Veamos a continuación algunas conclusiones. A pesar de que en el momento de desarrollar los presentes materiales el negocio basado en software libre es relativamente reciente, se han puesto de relieve las diferencias que permiten una nueva perspectiva del negocio basada principalmente en la potenciación de la producción cooperativa del conocimiento.

La aplicación basada en software libre

La producción social de una aplicación o solución concreta favorece la creación de valor por encima de su coste de producción, lo que confiere una ventaja competitiva respecto a otras alternativas del mercado.

Las aplicaciones basadas en software libre, conjuntamente con los estándares abiertos, pueden contrarrestar algunos de los efectos económicos que fortalecen los productos basados en el modelo tradicional. En este sentido, además de inducir una diferenciación sustancial respecto las aplicaciones tradicionales, permiten estrategias y políticas de coopetencia entre empresas bajo un paradigma ganador-ganador.

El mercado

La producción social ha plagado Internet de iniciativas alternativas a los modelos tradicionales. El capital social ha devenido, con el paso del tiempo, un importante valor para la innovación y desarrollo en entornos abiertos. En la actualidad existen modelos de negocio lucrativos que remuneran la producción de conocimiento.

El negocio del conocimiento

Innocentive (http://www.innocentive.com/), entre otros, es un portal web que se dedica a recompensar las ideas que solucionan problemas concretos. Es decir, hay usuarios que plantean problemas (*seekers*) y otros que los solucionan (*solvers*) a cambio de una compensación económica.

Este y otros ejemplos han dado pie a la creación de una nueva lógica de mercado, llamada en algunos contextos wikinomía y *crowdsourcing*. Esta lógica se basa en el modelo *pull* que hemos visto anteriormente, es decir, la atracción de ideas y esfuerzos frente al tradicional modelo *push*.

Con el paso del tiempo veremos si esta perspectiva de mercado permite la evolución de los patrones de adopción tecnológica propios del mercado tradicional hacia una nueva situación.

El negocio

La nueva perspectiva de mercado puede ofrecer nuevas oportunidades de negocio ligadas a la explotación lucrativa de ideas, conceptos y conocimientos sin tener que ser propietarios de los mismos. Es decir, el valor de la aplicación basada en software libre no está en la solución en sí, sino en el capital que se adquiere y en el que se puede generar con él.

Aun así, la validez y la viabilidad del software libre como modelo también se sustentan en las particularidades de concepción de la empresa que lo explota. Es decir, resulta fundamental concebir la empresa sobre una oportunidad de negocio sólida y duradera.

Los riesgos

Sin duda, los principales riesgos para el modelo basado en software libre son obtener una masa crítica de usuarios que permita la viabilidad del proyecto y fundamentar un modelo de negocio estable a lo largo del tiempo. También hay que tener en cuenta la relación entre inversión inicial y beneficios esperados.

El estudio de viabilidad empresarial

Analizar, diseñar y formalizar exhaustivamente la empresa nos permite aumentar las garantías de éxito del negocio basado en software libre. Para maximizar estas garantías, la viabilidad de la empresa debe estudiarse previamente a su puesta en marcha y formalizarse en un plan de empresa.

Ved también

En el tercer módulo de esta asignatura ya se ha realizado una primera aproximación a las principales características ligadas a la viabilidad empresarial del negocio clásico del software, por ejemplo los aspectos de comercialización y de marketing, así como a los productos y servicios objeto del negocio empresarial.

En la empresa basada en software libre debemos complementar los aspectos anteriores con las características de los modelos de negocio basados en software libre vistos en el cuarto módulo, de forma que la combinación permita formalizar una base sólida sobre la que fundamentar un negocio sostenible.

La empresa de software libre

Como en cualquier otro modelo empresarial, la empresa basada en el software libre también requiere un diseño y planificación detallados antes de su puesta en marcha. En los anteriores apartados hemos puesto de relieve la importancia de analizar detenidamente los fundamentos comerciales de la empresa como condición para evaluar su validez y viabilidad.

Tanto los fundamentos del software libre como las implicaciones que hemos ido detallando a lo largo del presente módulo pueden ejercer fuerzas diferentes en función de la tipología de la oportunidad de negocio y el contexto donde se pretende explotar.

De este modo, la estrategia de la empresa basada en software libre puede y debe caracterizar sus actuaciones en la diferenciación del negocio y en los efectos económicos de su entorno, así como en el capital y producción social, además de la coopetencia.

Resumen

A lo largo de este módulo hemos examinado las características del software libre como modelo económico, aun considerando las limitaciones propias de la escasez de datos debido a que los modelos de negocio basados en software libre son relativamente recientes.

Por una parte, los fundamentos del capital social y la producción colectiva de ideas y conocimientos no son exclusivos del software libre. En la actualidad existen diferentes iniciativas que demuestran cuán factibles pueden resultar la cooperación y colaboración en la innovación y producción de conocimientos.

Estos fundamentos denotan la importancia de la red de colaboradores, su implicación y motivación en el avance global e individual de los miembros de la comunidad, y suponen una alternativa viable a los modelos tradicionales de producción.

Por otra parte, las implicaciones de la filosofía de producción social pueden ser examinadas desde diferentes puntos de vista. Si bien es cierto que ciertas características del software libre no suponen una diferencia significativa con respecto a otros modelos, existen otras características que sí promueven una diferenciación importante.

Considerando el negocio del software libre, resulta primordial fortalecer y explotar los fundamentos diferenciadores del software libre para ofrecer alternativas válidas y viables a los modelos tradicionales. Estas actuaciones deben complementarse necesariamente con el estudio y planificación detallados de la oportunidad de negocio para garantizar la viabilidad presente y futura de la empresa del software libre.

Bibliografía

Benkler, Y. (2006). *The wealth of networks: How social production reforms markets and freedom*. New Haven: Yale University Press. <http://www.benkler.org/Benkler_Wealth_Of_Networks.pdf; WIKI:http://cyber.law.harvard.edu/wealth_of_networks/Main_Page> [Fecha de consulta: marzo del 2009]

Bollier, D. (2006). *When Push Comes to Pull: The New Economy and Culture of Networking Technology.* <http://www.aspeninstitute.org/atf/cf/%7bDEB6F227-659B-4EC8-8F84-8DF23CA704F5%7d/2005InfoTechText.pdf>

Fogel, K. (2004). *The Promise of the Post-Copyright World* <http://www.questioncopyright.org/promise> [Fecha de consulta: marzo del 2009]

Fuggetta, A. (sept.-oct., 2004): *Open Source and Free Software: A New Model for the Software Development Process?* (núm. 171). Novática – Upgrade: Monografía del proceso de software, Inglés <http://www.upgrade-cepis.org/issues/2004/5/up5-5Fuggetta.pdf) | Español: (http://alarcos.inf-cr.uclm.es/doc/ig1/doc/temas/4/IG1-t4slibreabierto.pdf> [Fecha de consulta: febrero del 2009]

Goldhaber, M. (junio, 2006). *The Value of Openness in an Attention Economy* (vol. 11, núm. 6). <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1334/1254> [Fecha de consulta: marzo del 2009]

Moglen, E. (1999). *Anarchism Triumphant and the Death of Copyright*. <http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/684/594> [Fecha de consulta: marzo del 2009]

Morgan, L.; Finnegan, P. (2008). *Deciding on open innovation: an exploration of how firms create and capture value with open source software*. En: G. León; A. Bernardos; J. Casar; K. Kautz; J. DeGross (ed.). International Federation for Information Processing. Open IT-Based Innovation: Moving Towards Cooperative IT Transfer and Knowledge Diffusion (vol. 287, pág. 229-246). Boston: Springer.

O'Reilly, T. (2004). *Open Source Paradigm Shift.* <http://tim.oreilly.com/articles/paradigmshift_0504.html> [Fecha de consulta: febrero del 2009]

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in per-

forming those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered

work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of

the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license

document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>
```

```
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.