ieu
ONLINE

Semana 1
Software Libre

Lectura
Introducción al software libre

González Barahona, J. S. (2003). Introducción al software libre. Barcelona: Barcelona: Fundació per a la Universitat Oberta de Catalunya. Unidad 2: Un poco de historia (pp. 31 – 43). Unidad 4: El desarrollador y sus motivaciones (pp. 98 – 99). Unidad 5: Economía (pp. 103 – 129)

2. Un poco de historia

Aunque todas las historias relacionadas con la informática son forzosamente breves, la del software libre es una de las más largas entre ellas. De hecho, podría decirse que el software nació libre y permaneció así durante su infancia. Sin embargo, con la llegada de la juventud, la situación cambió completamente. Sólo ahora, al llegar a su madurez, está en vías de recuperar la libertad. Esto no deja de ser curioso, pues para gran parte de los profesionales informáticos, el software propietario es el software *en su estado natural*. Afortunadamente, la situación es más bien la contraria, y las semillas del cambio que estamos empezando a entrever en los últimos años fueron plantadas ya a principios de la década de 1980.

Sugerencia

No hay muchas historias más o menos exhaustivas sobre el software libre, y las que existen son artículos más o menos limitados en el objeto de su estudio. En cualquier caso, el lector interesado puede completar lo expuesto en este capítulo consultando "Open Source Initiative. History of the osi" (énfasis en el impacto en el mundo empresarial, entre los años 1998 y 1999) o "Chris Rasch. A brief history of free/open source software movement", que cubre la historia del software libre hasta el año 2000, o "Nathan Newman. The origins and future of open source software, 1999", que se centra en gran medida en la promoción indirecta que el Gobierno de EE.UU. ha hecho del software libre, o sistemas similares, durante las décadas de 1970 y 1980.

2.1. El software libre antes del software libre

El software libre como concepto no apareció hasta principios de la década de 1980. Sin embargo, su historia puede trazarse desde bastantes años antes.

2.1.1. Y en el principio fue libre...

Durante los años 1960 el panorama de la informática estaba dominado por los grandes ordenadores, instalados fundamentalmente en empresas y centros gubernamentales. IBM era el principal fabricante, con gran diferencia sobre sus competidores. En esta época, cuando se adquiría un ordenador (el hardware), el software venía como un acompañante. Mientras se pagase el contrato de mantenimiento, se tenía acceso al catálogo de software que ofrecía el fabricante. Además, no era común la idea de que los programas fueran algo *separado* desde un punto de vista comercial.

En esta época el software se distribuía habitualmente junto con su código fuente (en muchos casos sólo como código fuente), y en general sin restricciones prácticas. Los grupos de usuarios como SHARE (usuarios de sistemas IBM) o DECUS (usuarios de DEC) participaban, y hasta cierto punto organizaban, estos intercambios. La sección *Algorithms* de la revista *Communications of the ACM* era otro buen ejemplo de foro de intercambio. Podría decirse que durante estos primeros años de la informática el software era libre, al menos en el sentido de que los que tenían acceso a él podían disponer habitualmente del código fuente, estaban acostumbrados a compartirlo, a modificarlo y a compartir las modificaciones. En palabras de Richard Stallman, refiriéndose a la situación en el Massachusetts Institute of Technology, uno de los lugares con más penetración de uso y más prestigio en investigación sobre informática en la época:

> "No denominábamos software libre a nuestro software porque este término no existía, pero eso es lo que era. Cuando alguien de otra universidad o de una empresa deseaba portar y usar un programa, se lo permitíamos

> con gusto. Si veías a alguien usando un programa interesante y poco conocido, siempre podías pedir el código fuente para verlo, de manera que podías leerlo, cambiarlo, o canibalizar ciertas partes del mismo para hacer un nuevo programa."

El 30 de junio de 1969 IBM anunció que a comienzos de 1970 iba a empezar a vender parte de su software por separado. Esto supuso que sus clientes ya no podían obtener, incluido en el precio del hardware, los programas que necesitaban. El software se comenzó a percibir como algo con valor intrínseco y, como consecuencia, se hizo cada vez más habitual restringir escrupulosamente el acceso a los programas, y se limitaron, en la medida de lo posible (tanto técnica como legalmente), las posibilidades que tenían los usuarios para compartir, modificar o estudiar el software. En otras palabras, se pasó a la situación que aún hoy es la habitual en el mundo del software.

Sugerencia

El lector interesado en esta época de transición puede leer, por ejemplo, Larry Welke; Luanne Johnson (1998). *How the icp directory began*, donde Larry Welke comenta cómo nació uno de los primeros catálogos de software no ligados a un fabricante, y cómo en este proceso descubrió que las empresas estaban dispuestas a pagar por programas no creados por el fabricante de sus ordenadores.

A mediados de la década de 1970 era ya absolutamente habitual, en cualquier ámbito informático, encontrarse con software propietario. Esto supuso un gran cambio cultural entre los profesionales que trabajaban con software, a la vez que un gran número de empresas florecían en torno al nuevo negocio. Faltaba aún casi una década para que empezase a aparecer, de forma organizada y como reacción a esta situación, lo que hoy conocemos como software libre.

2.1.2. Años 1970 y primeros 1980

Incluso cuando la tendencia abrumadoramente mayoritaria era la de explorar el modelo de software propietario, había iniciativas que mostraban algunas características de lo que luego se consideraría software libre. Alguna de ellas incluso produjo software libre, según la definición que usamos hoy día. Entre ellas, caben destacar Spice y TeX, además del caso mucho más complejo de Unix.

Spice (Simulation Program with Integrated Circuit Emphasis) es un programa desarrollado en la Universidad de California en Berkeley para simular las características eléctricas de un circuito integrado. Fue desarrollado y puesto en el dominio público por su autor, Donald O. Pederson, en 1973. SPICE era originalmente una herramienta docente, y como tal se extendió rápidamente a muchas universidades de todo el mundo, por lo que fue usado por muchos estudiantes de la que por aquel entonces era una incipiente disciplina de diseño de circuitos integrados. Estando en el dominio público, SPICE podía redistribuirse, modificarse, estudiarse, etc. Incluso se podía adaptar a unas necesidades concretas y vender esa versión como producto propietario (lo que se ha hecho durante su historia docenas de veces por parte de una gran cantidad de empresas). Con estas características, SPICE tenía todas las papeletas para convertirse en el estándar de la industria con sus diferentes versiones. Y efectivamente, eso fue lo que ocurrió. Probablemente éste fue el primer programa con características de software libre que durante un tiempo monopolizó un mercado, el de los simuladores de circuitos integrados, y sin duda pudo hacerlo precisamente por tener estas características (además de sus innegables cualidades técnicas).

Donald Knuth comenzó a desarrollar TeX durante un año sabático, en 1978. TeX es un sistema de tipografía electrónica muy utilizado para la producción de documentos de calidad. Desde el principio, Knuth utilizó una licencia que hoy sería considerada como de software libre. Cuando el sistema se consideró razonablemente estable, en 1985, mantuvo esa licencia. En esa época, TeX era un de los sistemas más grandes y más conocidos que podía considerarse software libre.

2.1.3. Desarrollo temprano de Unix

Unix fue uno de los primeros sistemas operativos portables, creado originalmente por Thompson y Ritchie (entre otros) en los Bell Labs de AT&T. Su desarrollo ha continuado desde su nacimiento, hacia 1972, hasta hoy, con innumerables variantes comercializadas por literalmente decenas de empresas.

Durante los años 1973 y 1974, Unix llegó a muchas universidades y centros de investigación de todo el mundo, con una licencia que permitía su uso para fines académicos. Aunque había ciertas restricciones que impedían su distribución libre, entre las organizaciones que disponían de la licencia el funcionamiento fue muy similar al que se vio más tarde en muchas comunidades de software libre. Los que tenían acceso al fuente de Unix tuvieron un sistema que podían estudiar, mejorar y ampliar. Juntamente con él apareció una comunidad de desarrolladores, que pronto empezó a girar en torno al CSRG de la Universidad de California en Berkeley, y que desarrolló su propia cultura, que fue muy importante, como veremos más adelante, en la historia del software libre. Unix fue, hasta cierto punto, un ensayo temprano de lo que se vio con GNU y Linux varios años más tarde. Estaba confinado a una comunidad mucho más pequeña, y era necesaria la licencia de AT&T, pero en otros aspectos su desarrollo fue similar (en un mundo mucho menos comunicado).

Con el tiempo, Unix fue también un ejemplo temprano de los problemas que podían presentar los sistemas propietarios que a primera vista *tenían alguna característica del software libre*. Durante el final de la década de 1970, y sobre todo durante la de 1980, AT&T cambió su política, y el acceso a nuevas versiones de Unix se convirtió en algo difícil y caro. La filosofía de los primeros años, que hizo tan popular a Unix entre los desarrolladores, cambió radicalmente, hasta el punto de que en 1991 AT&T puso una demanda a la Universidad de Berkeley por publicar el código de Unix BSD que ellos (el CSRG de Berkeley) habían creado. Pero esa es otra historia, que retomaremos más adelante.

2.2. El comienzo: BSD, GNU

Todos los escenarios descritos hasta ahora son, o bien iniciativas individuales, o bien no cumplen los requisitos del software libre. En cualquier caso, hasta principios de la década de 1980 no aparecieron, de forma organizada y consciente, los primeros proyectos para la creación de sistemas compuestos de software libre, y lo que probablemente es más importante: los fundamentos éticos, legales y hasta económicos, que luego se continuarían desarrollando hasta el día de hoy. De esta época procede también el propio término *software libre*.

2.2.1. Richard Stallman, GNU, FSF: nace el movimiento del software libre

A principios de 1984, Richard Stallman, en aquella época empleado en el AI Lab del MIT, abandonó su trabajo para comenzar el proyecto GNU. Stallman se consideraba un *hacker* de los que gozaban compartiendo sus inquietudes tecnológicas y su código. Veía con desagrado cómo su negativa a firmar acuerdos de exclusividad y no compartición le estaban convirtiendo en un extraño en su propio mundo, y cómo el uso de software propietario en su entorno le dejaba impotente antes situaciones que antes podía solventar fácilmente.

Su idea al abandonar el MIT era construir un sistema de software completo, de propósito general, pero completamente libre. El sistema (y el proyecto que se encargaría de hacerlo realidad) se llamó GNU (acrónimo recursivo, *GNU's Not Unix*). Aunque desde el principio el proyecto GNU incluyó en su sistema software ya disponible (como Tex, o más adelante, el sistema X Window), había mucho que construir. Richard Stallman comenzó por escribir un compilador de C (GCC) y un editor (Emacs), ambos aún en uso (y muy populares) hoy día.

Desde el principio del proyecto GNU, Richard Stallman estaba preocupado por las libertades que tendrían los usuarios de su software. Estaba interesado en que no sólo los que recibieran los programas directamente del proyecto GNU, sino cualquiera que lo recibiera después de cualquier número de redistribuciones y (quizás) modificaciones, siguiera disfrutando de los mismos derechos (modificación,

redistribución, etc.). Para ello, escribió la licencia GPL, probablemente la primera licencia de software diseñada específicamente para garantizar que un programa fuera libre en este sentido. Al mecanismo genérico que utilizan las licencias tipo GPL para conseguir estas garantías, Richard Stallman lo llamó *copyleft*, que hoy día es el nombre de una gran familia de licencias de software libre.

Richard Stallman también fundó la Free Software Foundation (FSF con el fin de conseguir fondos para el desarrollo y la protección del software libre, y sentó los fundamentos éticos del software libre, con documentos como *The GNU Manifesto* y *Why Software Should Not Have Owners*.

Desde el punto de vista técnico, el proyecto GNU fue concebido como un trabajo muy estructurado y con metas muy claras. El método habitual estaba basado en grupos relativamente pequeños de personas (habitualmente voluntarios) que desarrollaban alguna de las herramientas que luego encajarían perfectamente en el rompecabezas completo (el sistema GNU). La modularidad de UNIX, en la que se inspiraba el desarrollo, encajaba perfectamente en esta idea. El método de trabajo generalmente implicaba el uso de Internet, pero ante la escasa implantación de aquellos días, la Free Software Foundation también vendía cintas en las que grababa las aplicaciones, siendo probablemente uno de las primeras organizaciones en beneficiarse económicamente (aunque de manera bastante limitada) de la creación de software libre.

A principios de la década de 1990, unos seis años después de su nacimiento, el proyecto GNU estaba muy cerca de tener un sistema completo similar a Unix. Aun así, hasta ese momento todavía no había producido una de las piezas fundamentales: el kernel del sistema (el núcleo del sistema operativo que se relaciona con el hardware y permite que todo funcione). Sin embargo, el software de GNU era muy popular entre los usuarios de las distintas variantes de Unix, por aquella época el sistema operativo más usado en las empresas. Además, el proyecto GNU había conseguido ser relativamente conocido entre los profesionales informáticos, y muy especialmente entre los que trabajaban en universidades. En esa época, sus productos ya gozaban de una merecida reputación de estabilidad y calidad.

2.2.2. El CSRG de Berkeley

El CSRG (Computer Science Research Group) de la Universidad de California en Berkeley fue, desde 1973, uno de los centros donde más se desarrolló todo lo relacionado con Unix durante los años 1979 y 1980. No sólo se portaron aplicaciones y se construyeron otras nuevas para su funcionamiento sobre Unix, sino que se hicieron importantes mejoras al kernel, y se le añadió mucha funcionalidad. Por ejemplo, durante la década de los años 1980, varios contratos de DARPA (dependiente del Ministerio de Defensa de EE.UU.) financiaron la implementación de TCP/IP que ha sido considerada hasta nuestros días como la referencia de los protocolos que hacen funcionar Internet (vinculando, de paso, el desarrollo de Internet y la expansión de las estaciones de trabajo con Unix). Muchas empresas utilizaron como base de sus versiones de Unix los desarrollos del CSRG, dando lugar a sistemas muy conocidos en la época, como SunOS (Sun Microsystems) o Ultrix (Digital Equipment). De esta manera, Berkeley se convirtió en una de las dos fuentes fundamentales de Unix, junto con la *oficial*, AT&T.

Para poder utilizar todo el código que producía el CSRG (y el de los colaboradores de la comunidad Unix que ellos de alguna manera coordinaban), hacía falta la licencia de Unix de AT&T, que cada vez era más difícil (y más cara) de conseguir, sobre todo si se quería el acceso al código fuente del sistema. Tratando de evitar en parte este problema, en junio de 1989 el CSRG liberó la parte de Unix relacionada con TCP/IP (la implementación de los protocolos en el kernel y las utilidades), que no incluía código de AT&T. Fue la llamada *Networking Release 1* (Net-1). La licencia con que se liberó fue la famosa *licencia BSD* que, salvo ciertos problemas con sus cláusulas sobre obligaciones de anuncio, ha sido considerada siempre como un ejemplo de licencia libre minimalista (que además de permitir la redistribución libre, también permitía su incorporación a productos propietarios). Además, el CSRG probó un novedoso método de financiación (que ya estaba probando con éxito la FSF): vendía cintas con su distribución por 1.000 dólares. Aunque cualquiera podía a su vez redistribuir el contenido de las cintas sin ningún problema (la licencia lo permitía), el CSRG vendió cintas a miles de organizaciones, con lo que consiguió fondos para seguir desarrollando.

Dado el éxito de la distribución de Net-1, Keith Bostic propuso reescribir todo el código que aún quedaba del Unix original de AT&T. A pesar del escepticismo de algunos miembros del CSRG, realizó un anuncio público pidiendo ayuda para realizar esta tarea, y poco a poco las utilidades reescritas a partir de especificaciones comenzaron a llegar a Berkeley. Mientras tanto, el mismo proceso se realizó con el kernel, y se reescribió de forma independiente la mayor parte del código que no se había realizado por parte de Berkeley o por colaboradores voluntarios. En junio de 1991, y después de conseguir el permiso de la Administración de la Universidad de Berkeley, se distribuyó la *Networking Release 2* (Net-2), con casi todo el código del kernel y todas las utilidades de un sistema Unix completo. De nuevo, todo esto se distribuyó bajo la licencia BSD, y de nuevo se vendieron miles de cintas al precio de 1.000 dólares la unidad.

Sólo seis meses después de la liberación de Net-2, Bill Jolitz escribió el código que faltaba en el kernel para que funcionase sobre arquitectura i386, liberando 386BSD, que fue distribuida por Internet. A partir de este código surgieron, en sucesión, todos los sistemas de la familia *BSD. Primero apareció NetBSD, como una recopilación de los parches que se habían ido portando en la Red para mejorar 386BSD. Más adelante apareció FreeBSD como un intento de soportar fundamentalmente la arquitectura i386. Varios años más tarde se formó el proyecto OpenBSD, con énfasis en la seguridad. Y también hubo una distribución propietaria a partir de Net-2 (aunque era ciertamente original, ya que ofrecía a sus clientes todo el código fuente como parte de la distribución básica), realizada de forma independiente por BSDI.

En parte como reacción a la distribución hecha por BSDI, Unix System Laboratories (USL), subsidiaria de AT&T que tenía los derechos de la licencia de Unix, puso una demanda judicial, primero a BSDI y luego a la Universidad de California. En ella les acusaba de distribuir su propiedad intelectual sin permiso. Después de varias maniobras judiciales (incluyendo una contrademanda de la Universidad de California contra USL), Novell compró los derechos de Unix a USL, y en enero de 1994 llegó a un acuerdo extrajudicial con la Universidad de California. Como resultado de este acuerdo, el CSRG distribuyó la versión 4.4BSD-Lite, que fue pron-

to utilizada por todos los proyectos de la familia *BSD. Poco después (tras liberar aún la versión 4.4BSD-Lite Release 2), el CSRG desapareció. En ese momento hubo quien temió que ello representara el fin de los sistemas *BSD, pero el tiempo ha demostrado que siguen vivos y coleando, con una nueva forma de gestión, más típicamente de proyectos libres. Hoy día, los proyectos que gestionan la familia *BSD son de los más antiguos y consolidados en el mundo del software libre.

Sugerencia

La historia del desarrollo de Unix BSD, desde sus orígenes en una cinta que llevó Bob Fabry a Berkeley con una de las primeras versiones del código de Thompson y Ritchie para hacerla funcionar en un PDP-11 que compraron conjuntamente los departamentos de informática, estadística y matemáticas, hasta las demandas judiciales de AT&T y las últimas liberaciones de código que dieron lugar a la familia de sistemas operativos libres *BSD es muy ilustrativa de una cierta forma de desarrollar software durante los años 1970 y 1980. Quien esté interesado en ella puede disfrutar de la lectura de Marshall Kirk McKusick. "Twenty Years of Berkeley Unix. From AT&T-Owned to Freely Redistributable". En: DiBona et al. (1999), donde se comentan muchos detalles de esta época.

2.2.3. Los comienzos de Internet

Casi desde su nacimiento, a principios de la década de 1970, Internet tuvo mucha relación con el software libre. Por un lado, desde sus comienzos, la comunidad de desarrolladores que la construyeron tuvieron claros varios principios que luego se harían clásicos en el mundo del software libre. Por ejemplo, la importancia de dar posibilidades a los usuarios para que ayuden a depurar los errores, o la compartición del código de las implementaciones. La importancia de BSD Unix en su desarrollo (al proporcionar durante los años 1980 la implementación más popular de los protocolos TCP/IP) hizo que mu-

chas costumbres y formas de funcionamiento pasasen fácilmente de una comunidad (la de desarrolladores alrededor del CSRG) a otra (los que estaban construyendo NSFNet, y luego simplemente Internet), y viceversa. Muchas de las aplicaciones básicas en el desarrollo de Internet, como Sendmail (servidor de correo) o Bind (implementación del servicio de nombres) fueron libres, y en gran medida fruto de esta colaboración entre comunidades.

Por último, la comunidad del software libre, durante el final de los años 1980 y la década de 1990, fue una de las primeras en explorar hasta el fondo las nuevas posibilidades que permitía Internet para la colaboración entre grupos geográficamente dispersos. Esta exploración fue la que hizo posible, en gran medida, la propia existencia de la comunidad BSD, la FSF o el desarrollo de GNU/Linux.

Uno de los aspectos más interesantes del desarrollo de Internet, desde el punto de vista del software libre, fue la gestión completamente abierta de sus documentos y sus normas. Aunque hoy pueda parecer algo normal (pues es lo habitual, por ejemplo, en el IETF o el WWW Consortium), en su época la libre disposición de todas las especificaciones y documentos de diseño, incluyendo las normas que definen los protocolos, fue algo revolucionario y fundamental para su desarrollo. Podemos leer en Michael Hauben and Ronda Hauben. (1997). *Netizens. On the History and Impact of Usenet and the Internet* (pág. 106). IEEE Computer Society Press.

> "Este proceso abierto promovía y llevaba al intercambio de información. El desarrollo técnico tiene éxito sólo cuando se permite que la información fluya libre y fácilmente entre las partes interesadas. La promoción de la participación es el principio fundamental que hizo posible el desarrollo de la Red."

Observe el lector cómo este párrafo podría ser suscrito, casi con toda seguridad, por cualquier desarrollador refiriéndose al proyecto de software libre en el que colabora.

En otra cita, Lawrence Roberts. "The evolution of packet switching". *Proceedings of the IEEE* (vol.66, pág. 267) podemos leer:

> "Como ARPANET era un proyecto público que conectaba muchas de las principales universidades e instituciones de investigación, los detalles de implementación y rendimiento se publicaban ampliamente."

Obviamente, esto es lo que ocurre hoy día en los proyectos de software libre, donde toda la información relacionada con el proyecto (y no sólo la implementación) suele ser pública.

En este ambiente, y hasta que Internet ya bien entrados los años 1990 se convirtió sobre todo en un negocio, la comunidad de usuarios y su relación con los desarrolladores era clave. En esa época muchas organizaciones aprendieron a confiar no en una única empresa proveedora del servicio de comunicación de datos, sino en una compleja combinación de empresas de servicios, fabricantes de equipos, desarrolladores profesionales y voluntarios, etc. Las mejores implementaciones de muchos programas no eran las que venían con el sistema operativo que se compraba con el hardware, sino implementaciones libres que rápidamente las sustituían. Los desarrollos más innovadores eran fruto no de grandes planes de investigación en empresas, sino de estudiantes o profesionales probando ideas y recogiendo la realimentación que les enviaban cientos de usuarios que usaban sus programas libres.

Como ya se ha dicho, Internet también proporcionó al software libre las herramientas básicas para colaborar a distancia. El correo electrónico, los grupos de News, los servicios de FTP anónimo (que fueron los primeros almacenes masivos de software libre), y más tarde los sistemas de desarrollo integrados basados en web han sido fundamentales (e imprescindibles) para el desarrollo de la comunidad del software libre tal como la conocemos, y en particular para el funcionamiento de la inmensa mayoría de los proyectos de software libre. Desde el principio, proyectos como GNU o BSD hicieron un uso masivo e intenso de todos estos mecanismos, desarrollando, a la vez

que las usaban, nuevas herramientas y sistemas que a su vez mejoraban Internet.

Sugerencia

El lector interesado en una historia de la evolución de Internet, escrita por varios de sus protagonistas, puede consultar Barry M. Leiner; Vinton G. Cerf; Robert E. Kahn; David D. Clark; Leonard Kleinrock; Daniel C. Lynch; Jon Postel; Larry G. Roberts; Stephen Wolff (1997). *A brief history of the intert. Communications of the ACM.*

2.2.4. Otros proyectos

Durante la década de 1980 vieron la luz otros importantes proyectos libres. Entre ellos destaca, por su importancia y proyección futura, el sistema X Window (sistema de ventanas para sistemas tipo Unix), desarrollado en el MIT, y que fue uno de los primeros ejemplos de financiación en gran escala de proyectos libres con recursos de un consorcio de empresas. También merece la pena mencionar Ghostscript, un sistema de gestión de documentos Postscript desarrollado por una empresa, Aladdin Software, que fue uno de los primeros casos de búsqueda de modelo de negocio produciendo software libre.

En general, y sobre todo a finales de los años 1980, están en marcha toda una serie de pequeños (y no tan pequeños) proyectos libres. Todos ellos, junto con los grandes proyectos mencionados hasta aquí, estaban ya sentando las bases de los primeros sistemas libres completos que aparecieron a principios de la década de 1990.

2.3. Todo en marcha

Hacia 1990, gran parte de los componentes de un sistema informático completo estaban ya listos como software libre. Por un lado, el proyecto GNU y por otro, las distribuciones BSD habían completado

Nota

Además de poder ver la profesionalización de los equipos de desarrollo de software libre, la dedicación en horas es un parámetro de gran importancia a la hora de poder realizar estimaciones de coste y hacer comparaciones con los modelos de desarrollo propietarios que se siguen en la industria. En el software libre, por ahora, sólo contamos con productos finales (nuevas entregas del software, sincronización de código nuevo en los sistemas de versiones) que no nos permiten conocer cuánto tiempo ha necesitado el desarrollador en conseguirlo.

El análisis de estas cifras nos muestra que en torno a un 80% de los desarrolladores realizan estas tareas en su tiempo libre, mientras que sólo uno de cada cinco podría considerarse como que dedica tanto tiempo a esta actividad como un profesional. Más adelante, en el capítulo de ingeniería del software podremos ver cómo este dato concuerda con la contribución de los desarrolladores, ya que ambas parecen seguir la ley de Pareto (ver apartado 7.6).

4.6. Motivaciones

Se ha especulado y se sigue especulando mucho sobre las motivaciones que hay detrás del desarrollo de software libre, sobre todo en su vertiente de actividad de tiempo libre (que, como hemos visto, corresponde a cerca de un 80% de los desarrolladores). Como en los apartados anteriores, sólo contamos con los datos de las encuestas, por lo que es importante darse cuenta de que se trata de lo que los desarrolladores responden, que puede ser más o menos coherente con la realidad. Los porcentajes que se van a mostrar a continuación superan en suma el 100%, ya que se daba la posibilidad a los encuestados de elegir varias respuestas.

En cualquier caso, de sus respuestas parece desprenderse que la mayoría quiere aprender y desarrollar nuevas habilidades (cerca de un 80%) y que muchos lo hacen para compartir conocimientos y habilidades (50%) o para participar en una nueva forma de cooperación (alrededor de un tercio). El primer dato no parece nada

sorprendente, habida cuenta de que un profesional con mayores conocimientos se encuentra más cotizado que uno que no los posee. El segundo dato, sin embargo, no es tan fácil de explicar e incluso parece ir en contra de la afirmación de Nikolai Bezroukov, que viene a decir que los líderes de los proyectos de software libre tienen a buena cuenta no compartir toda la información de la que poseen para perpetuar su poder. Mientras tanto, la tercera opción más frecuente es, sin lugar a dudas, fiel reflejo de que los propios desarrolladores se muestran entusiasmados por la forma en la que generalmente se crea el software libre; es difícil encontrar una industria en la que un grupo de voluntarios levemente organizados pueda plantar cara tecnológicamente a las grandes compañías del sector.

Mientras la teoría *clásica* para explicar los motivos por los que los desarrolladores de software libre se dedican a aportar en proyectos de software libre gira en torno a la reputación y a beneficios económicos indirectos a medio y largo plazo, parece que los propios desarrolladores no están de acuerdo con estas afirmaciones. Sólo un 5% de los encuestados responde que desarrolla software libre para ganar dinero, mientras que el número de ellos que lo hacen por obtener reputación asciende a un 9%, lejos de las respuestas que se han presentado en el párrafo anterior. En cualquier caso, parece que el estudio de las motivaciones que tienen los desarrolladores para entrar a formar parte de la comunidad del software libre es una de las tareas primordiales con las que se han de enfrentar sociólogos y psicólogos en los próximos tiempos.

4.7. Liderazgo

Reputación y liderazgo son dos características con las que se ha tratado de explicar el éxito del software libre y en especial el del modelo de bazar, tal y como se verá en el capítulo dedicado a la ingeniería del software. Como pudimos ver en otro capítulo, el dedicado a las licencias del software, existen ciertas diferencias entre las licencias de software libre y sus homólogas en el campo de la documentación. Esas diferencias radican en la forma como se preservan la autoría y la opinión del autor –más acentuada en textos que en programas.

5. Economía

En este capítulo se tratan algunos aspectos económicos relacionados con el software libre. Se comienza mostrando cómo se financian los proyectos de software libre (cuando efectivamente se financian, ya que en muchos casos se desarrollan únicamente con trabajo y recursos aportados voluntariamente). A continuación, se exponen los principales modelos de negocio que están poniendo en práctica las empresas relacionadas directamente con el software libre. El capítulo termina con un pequeño estudio sobre la relación entre el software libre y los monopolios en la industria del software.

5.1. Financiación de proyectos de software libre

El software libre se desarrolla de muchas formas distintas, y con mecanismos para conseguir recursos que varían muchísimo de un caso a otro. Cada proyecto libre tiene su propia forma de financiarse, desde el que está formado completamente por desarrolladores voluntarios y utiliza solamente recursos cedidos altruistamente, hasta el que es llevado a cabo por una empresa que factura el 100% de sus costes a una entidad interesada en el desarrollo correspondiente.

En este apartado nos vamos a centrar en los proyectos donde hay financiación externa, y no todo el trabajo realizado es voluntario. En estos casos, hay algún tipo de flujo de capital con origen externo al proyecto que se encarga de aportar recursos para su desarrollo. De esta manera, el software libre construido puede considerarse, de alguna manera, como un producto de esta financiación externa. Por ello, es común que sea esa fuente externa quien decide (al menos parcialmente) cómo y en qué se gastan los recursos.

En cierto sentido, esta financiación externa para proyectos libres puede considerarse como un tipo de patrocinio, aunque este patrocino no tiene por qué ser desinteresado (y habitualmente no lo es). En los siguientes apartados se comentan los tipos de financiación externa más habituales. Mientras el lector se dedique a ellos, conviene, no

ANOTACIONES

obstante, no olvidar que ésta es sólo una de las maneras que tienen los proyectos que construyen software libre de conseguir recursos. Hay otras, y entre ellas la más importante: el trabajo de muchos desarrolladores voluntarios (como se discute en el capítulo 4).

5.1.1. Financiación pública

Un tipo muy especial de financiación de proyectos libres es la pública. La entidad financiadora puede ser directamente un gobierno (local, regional, nacional o incluso supranacional) o una institución pública (por ejemplo, una fundación). En estos casos, la financiación suele ser similar a la de los proyectos de investigación y desarrollo, y de hecho es muy habitual que provenga de entidades públicas promotoras de I+D. Normalmente, la entidad financiadora no busca recuperar la inversión (o al menos no de forma directa), aunque suele tener objetivos claros (favorecer la creación de tejido industrial e investigador, promover cierta tecnología o cierto tipo de aplicaciones, etc.).

En la mayor parte de estos casos, no se encuentra explícitamente la financiación de productos o servicios relacionados con software libre, sino que suelen ser el subproducto de un contrato con otros fines más generales. Por ejemplo, la Comisión Europea, dentro de sus programas de investigación, financia proyectos orientados a mejorar la competitividad europea en ciertas áreas. Algunos de estos proyectos tienen como parte de sus objetivos usar, mejorar y crear software libre en su ámbito de investigación (como herramienta para la investigación, o como producto derivado de ella).

Las motivaciones para este tipo de financiación son muy variadas, pero pueden destacarse las siguientes:

- Científica

Éste es el caso más habitual en proyectos de investigación financiados con fondos públicos. Aunque su objetivo no es producir software, sino investigar en un determinado campo (relacionado o no con la informática), es muy posible que para ello sea preciso desarrollar programas que se usen como herramientas necesarias

ANOTACIONES

para alcanzar las metas del proyecto. Normalmente, el proyecto no está interesado en comercializar estas herramientas, o incluso está activamente interesado en que otros grupos las utilicen y mejoren. En estos casos, es bastante habitual distribuirlos como software libre. Cabe mencionar que los recursos que consiguió el grupo que realiza la investigación se dedicaron en parte a la producción de este software, por lo que se puede decir que fue desarrollado con financiación pública.

- Promoción de estándares

Tener una implementación de referencia es una de las mejores formas de promover un estándar. En muchos casos eso supone tener programas que formen parte de esa implementación (o si el estándar se refiere al campo del software, que sean la implementación ellos mismos). Para que la implementación de referencia sea de utilidad en la promoción del estándar, es preciso que esté disponible, al menos para comprobar su interoperabilidad entre todos los que quieran desarrollar productos que se acojan a ese estándar. En muchos casos es conveniente también que los fabricantes puedan directamente adaptar la implementación de referencia para usarla en sus productos, si así lo desean. De esta manera se desarrollaron, por ejemplo, muchos de los protocolos de Internet que hoy se han convertido en norma universal. En estos casos, la liberación de esas implementaciones de referencia como software libre puede ayudar mucho en esa promoción, ya que el software libre es un subproducto de la promoción del estándar, y habitualmente, quien se encarga de esta promoción es una entidad pública (aunque a veces es un consorcio privado).

- Social

El software libre es una herramienta de gran interés en la creación de la infraestructura básica para la sociedad de la información. Las entidades interesadas en utilizar software libre para ayudar al acceso universal a esta sociedad de la información pueden financiar proyectos relacionados con el software libre (normalmente proyectos de desarrollo de nuevas aplicaciones o de adaptación de otras existentes).

Nota

Un ejemplo de financiación pública con finalidad fundamentalmente social es el caso de LinEx, promovido por la Junta de Extremadura (Extremadura, España) para promover la sociedad de la información fundamentalmente en lo que a alfabetización informática se refiere. La Junta ha financiado el desarrollo de una distribución basada en Debian para conseguir este fin. Otro caso similar es el de la financiación por parte del Gobierno alemán de desarrollos de GnuPG orientados a facilitar su uso a los usuarios no experimentados, con la idea de favorecer el uso del correo seguro entre sus ciudadanos.

Ejemplo

El desarrollo de GNAT

Un caso notorio de financiación pública para el desarrollo de software libre fue el del compilador GNAT. GNAT, compilador de Ada, fue financiado por el proyecto Ada9X del Departamento de Defensa de EE.UU., con la idea de disponer de un compilador de la nueva versión del lenguaje de programación Ada (la que luego fue Ada95), cuyo uso trataba de promover en aquella época. Una de las causas que se habían identificado en cuanto a la adopción de la primera versión de Ada (Ada83) por las empresas de software era la tardía disposición de un compilador del lenguaje, y su alto precio cuando estuvo finalmente disponible. Por ello, trataron de que no ocurriese lo mismo con Ada95, asegurándose de que hubiera un compilador de forma prácticamente simultánea con la publicación del nuevo estándar del lenguaje.

Para lograrlo, el proyecto Ada9X contrató un proyecto con un equipo de la Universidad de Nueva York (NYU), por un importe aproximado de 1 millón de USD, para la realización de una "prueba de concepto" de compilador de Ada95. Con estos fondos, y aprovechando la existencia de GCC (el compilador de C de GNU, del que se aprovechó la mayor parte del dorsal), el equipo de NYU construyó efectivamente el primer compilador

ANOTACIONES

de Ada95, que liberó bajo la GNU GPL. El compilador tuvo tanto éxito que al finalizar el proyecto parte de sus constructores fundaron una empresa (Ada Core Tecnologies), que desde entonces se ha convertido en líder en el mercado de compiladores y herramientas de ayuda a la construcción de programas en Ada.

Es notable cómo en este proyecto se puede ver la combinación de elementos de investigación (este proyecto avanzó en el conocimiento sobre la construcción de frontales y de sistemas de tiempo de ejecución para compiladores de lenguajes tipo Ada) y de promoción de estándares (que era el objetivo más claro de su financiador).

5.1.2. Financiación privada sin ánimo de lucro

Éste es un tipo de financiación con muchas características similares a las del caso anterior, donde la financiación la realizan normalmente fundaciones u organizaciones no gubernamentales. La motivación directa en estos casos suele ser producir software libre para su uso en algún ámbito que la entidad financiadora considera especialmente relevante, pero también puede encontrarse la motivación indirecta de contribuir a resolver un problema (por ejemplo, una fundación dedicada a promover la investigación sobre una enfermedad puede financiar la construcción de un programa estadístico que ayude al análisis de grupos de experimentación donde se está estudiando esa enfermedad).

En general, tanto los motivos para realizar esta financiación como sus mecanismos son muy similares a los de financiación pública, aunque naturalmente están siempre marcados por los objetivos de la entidad que financia.

Nota

Probablemente el caso paradigmático de fundación que promueve el desarrollo de software libre sea la Free Software Foundation (FSF). Desde mediados de la década de 1980 esta fundación se dedica a la promoción del proyecto GNU, y a fomentar en general el desarrollo del software libre.

Otro caso interesante, aunque en otro ámbito bastante diferente, es la Open Bioinformatics Foundation. Entre los fines de esta fundación se encuentra la de promover el desarrollo de los programas informáticos que son básicos para investigación en cualquiera de las ramas de la bioinformática. Y en general, realiza esta promoción financiando y ayudando a la construcción de programas libres.

5.1.3. Financiación por quien necesita mejoras

Otro tipo de financiación para el desarrollo de software libre, ya no tan altruista, es el que tiene lugar cuando alguien necesita mejoras en un producto libre. Por ejemplo, una empresa necesita para uso interno cierta funcionalidad en un programa dado. O bien necesita que ciertos errores en un programa sean corregidos. En este tipo de casos, lo habitual es que la empresa en cuestión contrate el desarrollo que necesita. Este desarrollo muy habitualmente (bien porque lo impone la licencia del programa modificado, bien porque la empresa lo decide así) es software libre.

Ejemplo

El caso de Corel y Wine

A finales de la década de 1990, Corel decidió portar sus productos a GNU/Linux. En el proceso de realizarlo, descubrió que un programa libre diseñado para facilitar la ejecución de binarios para Windows en entornos Linux podría permitirle muchos ahorros de desarrollo. Pero para ello era preciso mejorarlo, fundamentalmente añadiéndole la emulación de cierta funcionalidad de Windows que usaban los programas de Corel.

Para que se realizaran estas mejoras, Corel contrató a Macadamian, que contribuyó con sus mejoras al proyecto Wine. Con ello, tanto Corel como Wine salieron beneficiados.

ANOTACIONES

5.1.4. Financiación con beneficios relacionados

En este caso, lo que busca la entidad financiadora es conseguir beneficios en productos relacionados con el programa a cuyo desarrollo aporta recursos. Normalmente, en estos casos los beneficios que obtiene la empresa financiadora no son exclusivos, ya que otras pueden entrar también en el mercado de la venta de productos relacionados. Pero, o bien la cuota de mercado que tiene es suficiente como para que no le preocupe mucho repartir la tarta con otros, o bien tiene alguna ventaja competitiva clara.

Algunos ejemplos de productos relacionados con un software dado son:

- Libros

La empresa en cuestión vende manuales, guías de uso, textos para cursos, etc. relacionados con el programa libre que ayuda a financiar. Por supuesto otras empresas pueden vender también libros relacionados con aquél, pero normalmente el hecho de financiar el proyecto le proporcionará acceso a desarrolladores clave antes que a sus competidores, o simplemente conseguirá con ello una buena imagen de cara a la comunidad de usuarios del programa en cuestión.

- Hardware

Si una empresa financia el desarrollo de sistemas libres para cierto tipo de hardware, puede dedicarse a vender con más facilidad ese tipo de hardware. De nuevo, al ser libre el software desarrollado, pueden aparecer competidores que vendan aparatos del mismo tipo, usando esos desarrollos, pero sin colaborar a su financiación. Aun así, la empresa en cuestión dispone de varias ventajas sobre sus competidores, y una de ellas puede ser que su posición como aportadora de recursos para el proyecto le permite influir para conseguir que los desarrollos que se realicen con prioridad sean los que más le interesen.

- CD con programas

Probablemente el modelo de este tipo más conocido es el de las empresas que financian ciertos desarrollos que luego aplican a su dis-

tribución de software. Por ejemplo, tener un buen entorno de escritorio puede ayudar mucho a vender CD con una cierta distribución de GNU/Linux, y por tanto financiar su desarrollo puede ser un buen negocio para quien los vende.

Es importante darse cuenta de que, para estar en este apartado, la financiación en cuestión ha de hacerse con ánimo de lucro, y para ello la entidad financiadora debe percibir algún beneficio posible en esta financiación. En los casos reales, sin embargo, es habitual que haya siempre una combinación de ánimo de lucro y altruismo cuando una empresa aporta recursos para que se realice un programa libre del cual espera beneficiarse indirectamente.

Nota

Un caso muy conocido de aporte de recursos a un proyecto, si bien de forma relativamente indirecta, es la ayuda que la editorial O'Reilly presta al desarrollo de Perl. Naturalmente, no es casualidad que esa editorial sea también una de las principales editoras sobre temas relacionados con Perl. En cualquier caso, es obvio que O'Reilly no tiene la exclusiva de la edición de libros de ese tipo, y otras editoriales compiten en ese segmento de mercado, con diverso éxito.

VA Software (en sus comienzos VA Research, y más tarde VA Linux) ha colaborado activamente en el desarrollo del kernel de Linux. Con ello ha conseguido, entre otras cosas, colaborar a que su su continuidad esté asegurada, lo que era especialmente crítico para ella, de cara a sus clientes, cuando su principal negocio era vender equipos con GNU/Linux preinstalado.

Red Hat ha financiado el desarrollo de muchos componentes de GNOME, con lo que ha conseguido fundamentalmente tener un entorno de escritorio para su distribución, lo que ha contribuido a aumentar sus ventas. Como en otros casos, otros fabricantes de distribuciones se han beneficiado de este desarrollo, aunque muchos de ellos no hayan colaborado con el proyecto GNOME en la misma medida que Red Hat (y

ANOTACIONES

no son pocos los que no han colaborado en nada). A pesar de ello, Red Hat se beneficia de su contribución a GNOME.

5.1.5. Financiación como inversión interna

Hay empresas que, directamente como parte de su modelo de negocio, desarrollan software libre. Por ejemplo, una empresa puede decidir iniciar un nuevo proyecto libre en un ámbito donde percibe que puede haber oportunidades de negocio, con la idea de rentabilizar posteriormente esta inversión. Este modelo podría considerarse una variante del anterior (financiación indirecta), siendo los "beneficios relacionados" las ventajas que obtenga la empresa por la producción del programa libre. Pero, por ser en este caso el producto libre en sí mismo el que se espera que produzca los beneficios, parece conveniente abrir una clasificación específica para estos casos.

Este tipo de financiación da lugar a varios modelos de negocio. Cuando se analicen éstos –en el apartado 5.2– se explicarán también las ventajas que normalmente obtiene una empresa de esta inversión en un proyecto, y qué métodos suelen utilizarse para rentabilizarla. De todos modos, cabe destacar que en algunos casos puede que el software en cuestión se desarrolle simplemente para satisfacer las necesidades de la propia empresa, y que sólo después se decida liberar, y quizás abrir una línea de negocio relacionada con él.

Nota

Digital Creations (hoy Zope Corporation) es uno de los casos más conocidos de empresa que se dedica al desarrollo de software libre con la esperanza de rentabilizar su inversión. El proyecto libre en que más está invirtiendo es Zope, un servidor de aplicaciones que está teniendo un cierto éxito. Su historia con el software libre comenzó cuando la entonces Digital Creations buscaba capital-riesgo para desarrollar su servidor de aplicaciones propietario, hacia 1998. Uno de los grupos interesados en invertir en ellos (Opticality Ventures) les puso como condición que el producto resultante debía ser libre, porque

Nota

en caso contrario no veían cómo iba a poder conseguir una cuota de mercado significativa. Digital Creations se decidió por ese camino, y pocos meses después anunciaba la primera versión de Zope (unos años después cambió su nombre). Hoy día Zope Corporation está especializada en ofrecer servicios de consultoría, formación y soporte para sistemas de gestión de contenidos basados en Zope, y otros productos en los que sin duda Zope es la piedra angular.

Ximian (antes Helix Code) es un caso bien conocido de desarrollo de aplicaciones libres en entorno empresarial. Muy ligada desde sus orígenes al proyecto GNOME, Ximian ha producido sistemas software como Evolution (un gestor de información personal que tiene una funcionalidad relativamente similar a la ofrecida por MS Outlook), Red Carpet (un sistema fácil de usar para la gestión de paquetería en un sistema operativo) y Mono (una implementación de gran parte de .NET). La empresa fue fundada en octubre de 1999, y atrajo a muchos desarrolladores de GNOME que pasaron a formar parte de sus desarrolladores (en muchos casos, continuando su colaboración con el proyecto GNOME). Ximian se posicionó como una empresa de ingeniería experta en adaptaciones de GNOME, en la construcción de aplicaciones basadas en GNOME, y en general en proporcionar servicios de desarrollo basados en software libre, especialmente de herramientas relacionadas con el entorno de escritorio. En agosto del 2003, Ximian fue adquirida por Novell.

Cisco Enterprise Print System (CEPS) es un sistema de gestión de impresión para organizaciones con gran cantidad de impresoras. Fue desarrollado internamente en Cisco, para satisfacer sus propias necesidades, y liberado en el año 2000 bajo la GNU GPL. Es difícil estar seguro de los motivos que tuvo Cisco para hacer esto, pero es posible que tuviera que ver con la búsqueda de contribuciones externas (informes de

error, nuevos controladores, parches, etc.). En cualquier caso, lo que está claro es que, al no tener Cisco ningún plan para comercializar el producto y no estar muy claro su mercado potencial, no tenía mucho que perder con esta decisión.

5.1.6. Otros modos de financiación

Hay otros modos de financiación difíciles de clasificar entre los anteriores. Entre ellos pueden destacarse, a modo de ejemplo, los siguientes:

- Utilización de mercados para poner en contacto a desarrolladores y clientes

La idea que sostiene este modo de financiación es que, sobre todo para pequeños desarrollos, es difícil que un cliente que lo desea pueda entrar en contacto con un desarrollador que pueda acometerlo de forma eficiente. Para mejorar esta situación, se postulan los mercados de desarrollo de software libre, donde los desarrolladores publicarían sus habilidades y los clientes los desarrollos que precisan. Una vez un desarrollador y un cliente se han puesto de acuerdo, tenemos una situación similar a la ya descrita como "financiación por quien necesita las mejoras" en el apartado 5.1.3.

Ejemplo

SourceXchange

SourceXchange fue un ejemplo de mercado para poner en contacto a desarrolladores con sus clientes potenciales. Para ofrecer un proyecto, un cliente escribía una RFP (*request for proposal*, o 'petición de propuesta'), especificando qué desarrollo se necesita y cuántos recursos se está dispuesto a proporcionar para ese desarrollo. Estas RFP se publicaban en el sitio. Cuando un desarrollador veía una que le interesaba, hacía una oferta para ella. Cuando un desarrollador y un cliente se ponían de acuerdo en los términos del desarrollo,

comenzaba un proyecto. Normalmente, cada proyecto estaba supervisado por un *peer reviewer*, un revisor, que se encargaba de asegurarse de que el desarrollador cumplía las especificaciones, que efectivamente éstas tenían sentido, que aconsejaba sobre cómo llevar adelante el proyecto, etc. SourceXchange (propiedad de la empresa CollabNet) se encargaba de ofrecer el sitio, garantizar la competencia de los revisores, asegurarse del pago en caso de que los proyectos se completasen, y ofrecer herramientas para el seguimiento del proyecto (servicios que facturaba al cliente). El primer proyecto mediado por SourceXchange se terminó en marzo del 2000, pero poco más de un año después, en abril del 2001, el sitio cerró.

- Venta de bonos para financiar un proyecto

Esta idea de financiación es similar a la de los mercados de bonos a los que acuden las empresas, pero orientado al desarrollo de software libre. Presenta unas cuantas variantes, pero una de las más conocidas funciona como sigue. Cuando un desarrollador (individual o empresa) tiene idea de un nuevo programa o de una mejora a uno existente, lo escribe en forma de especificación, estima el coste que tendría su desarrollo y emite bonos para su construcción. Estos bonos tienen un valor que se ejecuta sólo si el proyecto finalmente se termina. Cuando el desarrollador ha vendido suficientes bonos, comienza el desarrollo, que va financiando con préstamos basados en ´éstos. Cuando termina el desarrollo y se certifica por alguna tercera parte independiente que efectivamente lo realizado cumple con las especificaciones, el desarrollador "ejecuta" los bonos que había vendido, paga sus deudas, y lo que le queda son los beneficios que obtiene por el desarrollo.

¿Quién estaría interesado en adquirir los mencionados bonos? Obviamente, los usuarios que desearan que ese nuevo programa o esa mejora a uno ya existente se realizaran. De alguna manera, este sistema de bonos permitiría que las partes interesadas fijaran (siquiera parcialmente) las prioridades de los desarrolladores mediante la compra de bonos. Ello permitiría al mismo tiempo que no hiciese fal-

ANOTACIONES

ta que una sola entidad asumiese los costes de desarrollo, sino que podrían repartirse entre muchas (incluyendo individuos), que además sólo tendrían que pagar si finalmente el proyecto termina con éxito. Un mecanismo muy similar a éste se propone, con mucho más detalle, en Chris Rasch. (2001). "The Wall Street performer protocol. using software completion bonds to fund open source software development". *First Monday*, (vol. 6, núm. 6, mayo).

Nota

El sistema de bonos descrito está basado en el *Street Performer Protocol* (protocolo del artista callejero), un mecanismo basado en el comercio electrónico diseñado para facilitar la financiación privada de trabajos de creación libres. Resumiendo, quien esté interesado en que se realice un determinado trabajo prometerá formalmente pagar una cierta cantidad si el trabajo se realiza y es publicado libremente. Sus intenciones son buscar una nueva manera de financiar trabajos relativamente pequeños que queden a disposición de todo el mundo, pero puede extenderse de muchas formas (los bonos para la construcción de software libre son una de ellas). Puede verse un pequeño caso de puesta en práctica de un derivado de este protocolo (el *Rational Street Performer Protocol*, Protocolo Racional del Artista Callejero, Paul Harrison en http://thecircle.org.au/funding_results.html, donde se aplica a la consecución de fondos para la financiación de parte de The Circle, un proyecto de software libre.

- Cooperativas de desarrolladores

En este caso, los desarrolladores de software libre, en lugar de trabajar individualmente o para una empresa, se reúnen en algún tipo de asociación (normalmente similar a una cooperativa). Por lo demás, su funcionamiento es similar al de una empresa, matizado quizás por su compromiso ético con el software libre, que puede ser parte de sus estatutos (aunque también una empresa puede hacer esto). En este tipo de organizaciones pueden darse combinaciones variadas de trabajo voluntario con trabajo remunerado. Un ejemplo de este tipo de organización es Free Developers.

5.2. Modelos de negocio basados en software libre

Además de los mecanismos de financiación de los proyectos, ya tratados, otro aspecto muy relacionado con la economía que merece la pena tratar es el de los modelos de negocio. Ya al hablar de estos mecanismos de financiación se han mencionado algunos brevemente, ahora en este apartado los describiremos de forma algo más metódica.

En general, puede decirse que son muchos los modelos de negocio que se están explorando en torno al software libre, algunos más clásicos y otros más innovadores. Hay que tener en cuenta que entre los modelos más habituales en la industria del software no es fácil usar aquellos basados en la venta de licencias de uso de programas producidos, pues en el mundo del software libre ése es un mecanismo de financiación muy difícil de explotar. Sin embargo, sí que se pueden utilizar los que están basados en el servicio a terceros, con la ventaja de que, sin ser necesariamente el productor de un programa, puede darse soporte completo sobre él.

Venta de software libre a tanto por copia

En el mundo del software libre es difícil cobrar licencias de uso, pero en general no imposible. Normalmente, no hay nada en las definiciones de software libre que impida que una empresa cree un producto y sólo se lo distribuya a quien le pague una cierta cantidad. Por ejemplo, un determinado productor podría decidir distribuir su producto con una licencia libre, pero sólo a quien le pague 1.000 euros por copia (de forma similar a como se hace en el mundo clásico del software propietario).

Sin embargo, aunque esto es teóricamente posible, en la práctica es bastante difícil de que suceda. Porque una vez que el productor ha *vendido* la primera copia, quien la recibe puede estar motivado a tratar de recu-

ANOTACIONES

perar su inversión vendiendo más copias a menor precio (algo que no puede prohibir la licencia del programa, si éste es libre). En el ejemplo anterior, podría tratar de vender 10 copias a 100 euros cada una, con lo que el producto le saldría gratis (y además dificultaría mucho que el productor original vendiese otra copia a 1.000 euros, si el producto puede obtenerse legalmente por la décima parte). Es fácil deducir cómo este proceso continuaría en cascada hasta la venta de copias a un precio cercano al coste marginal de copia, que con las tecnologías actuales es prácticamente cero.

Aun así, y teniendo en cuenta que el mecanismo descrito propiciará que normalmente el productor no pueda poner un precio (especialmente un precio alto) al simple hecho de la redistribución del programa, hay modelos de negocio que implícitamente hacen justamente eso. Un ejemplo es el de las distribuciones de GNU/Linux, que se venden por un precio muy bajo comparado con sus competidores propietarios, pero superior (y normalmente claramente superior) al coste de copia (incluso cuando se pueden bajar libremente de Internet). Por supuesto, en estos casos juegan además otros factores, como la imagen de marca o la comodidad para el consumidor. Pero no es éste el único caso. Por lo tanto, más que indicar que con software libre *no se puede vender a tanto por copia*, hay que tener en cuenta que es más difícil de hacer, y probablemente se obtendrá menos beneficio, pero que puede haber modelos basados justamente en eso.

Dadas estas limitaciones (y estas ventajas) desde hace unos años se están probando variantes de los modelos de negocio habituales en la industria que son viables con software, a la vez que se buscan modelos innovadores, que tratan de explotar las nuevas posibilidades que ofrece el software libre. Sin duda en los próximos años veremos aún más experimentación en este campo, y también tendremos más información sobre qué modelos pueden funcionar bien, y en qué circunstancias.

En este apartado vamos a ofrecer una panorámica de los que más habitualmente nos encontramos hoy día, agrupados con la intención de mostrar al lector lo que tienen en común y lo que les diferencia, centrándonos en los modelos basados en el desarrollo y los servicios en torno a un producto de software libre. Los ingresos en este caso provienen directamente de estas actividades de desarrollo y servicios para el producto, pero no necesariamente implican desarrollo de nuevos productos. Cuando sí que se hace desarrollo, estos modelos tienen como *subproducto* la financiación de productos de software libre, por lo que son modelos especialmente interesantes, y su impacto sobre el mundo del software libre en general puede ser grande.

En cualquier caso, y aunque aquí ofrecemos una clasificación relativamente estricta, no hay que olvidar que casi todas las empresas usan en realidad modelos mixtos, que son combinaciones de los modelos que hemos descrito, entre sí y con otros más tradicionales.

5.2.1. Mejor conocimiento

La empresa que utiliza este modelo de negocio trata de rentabilizar su conocimiento de un producto (o conjunto de productos) libres. Sus ingresos provendrán de clientes a los que venderá servicios relacionados con ese conocimiento, típicamente desarrollos basados en el producto, modificaciones, adaptaciones, instalaciones e integraciones con otros. La ventaja competitiva de la empresa estará en gran medida relacionada con el mejor conocimiento del producto en cuestión que el que tienen sus competidores: por ello, la empresa estará especialmente bien situada si es productora del producto o participa en el proyecto que lo produce.

Ésta es una de las razones por la que las empresas que utilizan este modelo suelen participar activamente en los proyectos relacionados con el software sobre el que tratan de vender servicios: es una forma muy eficiente de obtener conocimiento sobre él, y lo que es más importante, de que ese conocimiento sea reconocido. Desde luego, explicarle a un cliente que entre los empleados hay varios desarrolladores del proyecto que produce el software, que, por ejemplo, se quiere modificar, puede ser una buena garantía.

ANOTACIONES

Relación con los proyectos de desarrollo

Por lo tanto, este tipo de empresas tiene un gran interés en dar la imagen de que poseen un buen conocimiento de determinados productos libres. Una interesante consecuencia de ello es que su apoyo a proyectos de software libre (por ejemplo, participando activamente en ellos, o permitiendo a sus empleados que lo hagan durante su jornada laboral) no es por lo tanto algo meramente filantrópico. Por el contrario, puede ser uno de los activos más rentables de la empresa, ya que sus clientes lo valorarán muy positivamente como una muestra clara de que conocen el producto en cuestión. Además, de esta forma podrán seguir muy de cerca el desarrollo, tratando de asegurarse, por ejemplo, de que algunas de las mejoras demandadas por sus clientes pasan a formar parte del producto desarrollado por el proyecto.

Analizándolo desde un punto de vista más general, ésta es una situación donde ambas partes, la empresa y el proyecto de desarrollo, ganan con la colaboración que se lleva a cabo. El proyecto gana por el desarrollo realizado por la empresa, o porque algunos de sus desarrolladores pasan a ser remunerados (siquiera parcialmente) por su trabajo en el proyecto. La empresa gana en conocimiento del producto, en imagen hacia sus clientes y en una cierta influencia sobre el proyecto.

Los servicios que proporcionan este tipo de empresas pueden ser muy amplios, pero normalmente consisten en desarrollos a medida, adaptaciones o integraciones de los productos en los que son expertas, o bien servicios de consultoría, donde aconsejan a sus clientes cómo utilizar mejor el producto en cuestión (especialmente si es complejo, o si su correcto funcionamiento es crítico para el cliente en cuestión).

Ejemplo

Algunos ejemplos de empresas que hasta cierto punto utilizan este modelo de negocio son los siguientes:

- LinuxCare

Fundada en 1996, proporcionaba en sus orígenes servicios de consultoría y soporte para GNU/Linux y software libre en EE.UU., y su plantilla estaba compuesta fundamentalmente por expertos en GNU/Linux. Sin embargo, en 1992 cambió sus objetivos, y desde entonces se ha especializado en proporcionar servicios casi exclusivamente a Linux ejecutando sobre máquinas virtuales z/VM en grandes ordenadores de IBM. Su modelo de negocio ha cambiado también a *mejor conocimiento con limitaciones*, al ofrecer como parte fundamental de sus servicios una aplicación no libre, Levanta.

- Alcove

Fundada en 1997 en Francia, proporciona fundamentalmente servicios de consultoría, consultoría estratégica, soporte y desarrollo para software libre. Desde su fundación, Alcove ha mantenido en plantilla a desarrolladores de varios proyectos libres, y ha tratado de rentabilizarlo en términos de imagen. También ha tratado de ofrecer una imagen, en general, de empresa vinculada a la comunidad de software libre, por ejemplo, colaborando con asociaciones de usuarios y dando publicidad a sus colaboraciones con proyectos libres (por ejemplo, desde Alcove-Labs).

5.2.2. Mejor conocimiento con limitaciones

Estos modelos son similares a los expuestos en el apartado anterior, pero tratando de limitar la competencia a que pueden verse sometidos. Mientras que en los modelos *puros* basados en el mejor conocimiento cualquiera puede, en principio, entrar en competencia, ya

que el software utilizado es el mismo (y libre), en este caso se trata de evitar esa situación poniendo barreras a esa competencia. Estas barreras suelen consistir en patentes o licencias propietarias, que normalmente afectan a una parte pequeña (pero fundamental) del producto desarrollado. Por esta razón, estos modelos pueden considerarse en realidad como mixtos, en el sentido de que están a caballo entre el software libre y el propietario.

En muchos casos, la comunidad del software libre desarrolla su propia versión para ese componente, con lo que la ventaja competitiva puede desaparecer, o incluso volverse en contra de la empresa en cuestión si su *competidor* libre se convierte en el estándar del mercado y es demandado por sus propios clientes.

Ejemplo

Son muchos los casos donde se usa este modelo de negocio, ya que es común considerarlo como menos arriesgado que el de conocimiento *puro*. Sin embargo, las empresas que lo han usado han tenido evoluciones variadas. Algunas de ellas son:

- Caldera

La historia de Caldera es complicada. En sus inicios, creó su propia distribución de GNU/Linux, orientada a las empresas: Caldera OpenLinux. En el 2001, compró la división de Unix de SCO, y en el 2002 cambió su nombre a "SCO Group". Su estrategia empresarial ha dado tantos vuelcos como su nombre, desde su total apoyo a Linux hasta sus demandas contra IBM y Red Hat en el 2003, y su abandono de su propia distribución. Pero en lo que se refiere a este apartado, el negocio de Caldera, al menos hasta el 2002, es un claro exponente del mejor conocimiento con limitaciones. Caldera trataba de explotar su conocimiento de la plataforma GNU/Linux, pero limitando la competencia a que podía verse sometida mediante la inclusión de software propietario en su distribución. Esto hacía difícil a sus clientes cambiar de distribución una vez que la habían adoptado, pues aunque las demás distribuciones

de GNU/Linux incluían la parte libre de Caldera OpenLinux, no encontraban en ellas la parte propietaria.

- Ximian

Fundada en 1999 con el nombre de Helix Code por desarrolladores muy vinculados al proyecto GNOME, fue adquirida en agosto del 2003 por Novell. La mayor parte del software que ha desarrollado ha sido libre (en general, parte de GNOME). Sin embargo, en un ámbito muy concreto Ximian decidió licenciar un componente como software propietario: el Connector for Exchange. Éste es un módulo que permite a uno de sus productos estrella, Evolution (un gestor de información personal que incluye correo electrónico, agenda, calendario, etc.) interactuar con servidores MS Exchange, muy utilizados en grandes organizaciones. De esta manera, trata de competir ventajosamente con otras empresas que proporcionan servicios basados en GNOME, quizás con los productos desarrollados por la propia Ximian, que no podrán interaccionar tan fácilmente con Exchange. Salvo por este producto, el modelo de Ximian ha sido de *mejor conocimiento*, y también basado en ser la fuente de un programa (ver más adelante).

5.2.3. Fuente de un producto libre

Este modelo es similar al basado en el mejor conocimiento, pero especializándolo de forma que la empresa que lo utiliza es productora, casi en su totalidad, de un producto libre. Naturalmente, la ventaja competitiva aumenta al ser éstos los desarrolladores del producto en cuestión, controlar su evolución, y tenerlo antes que la competencia. Todo ello posiciona a la empresa desarrolladora en un lugar muy bueno de cara a los clientes que deseen servicios sobre ese programa. Además, es un modelo muy interesante en términos de imagen, ya que la empresa ha demostrado su potencial desarrollador con la creación y mantenimiento de la aplicación en cuestión, lo que puede ser muy interesante de cara

a convencer a posibles clientes de sus capacidades. Igualmente, proporciona muy buena imagen de cara a la comunidad del software libre en general, ya que reciben de la empresa un nuevo producto libre que pasa a formar parte del acervo común.

Ejemplo

Son muchos los productos libres que comenzaron su desarrollo dentro de una empresa, y muy habitualmente ha continuado siendo esa empresa quien ha guiado su desarrollo posterior. A modo de ejemplo, podemos citar los siguientes casos:

- Ximian

Ya hemos mencionado cómo, en parte, ha usado el modelo de mejor conocimiento con limitaciones. Pero, en general, Ximian ha seguido un claro modelo basado en ser la fuente de programas libres. Sus principales productos, como Evolution, RedCarpet o Mono se han distribuido bajo licencias de GNU, y Ximian los ha desarrollado casi en exclusiva desde el comienzo. La empresa ha tratado de rentabilizar este desarrollo consiguiendo contratos para hacerlos evolucionar en ciertos sentidos, para adaptarlos a las necesidades de sus clientes y ofreciendo personalización y mantenimiento.

- Zope Corporation

En 1995 se funda Digital Creations, que desarrolla un producto propietario para la gestión de anuncios clasificados vía web. En 1997 recibió una inyección de capital por parte, entre otros, de una empresa de capital-riesgo, Opticality Ventures. Lo extraño (en aquella época) de esta inversión es que le pusieron como condición que distribuyera como software libre la evolución de su producto, lo que más adelante fue Zope, uno de los gestores de contenidos más populares en Internet. El modelo de negocio de la empresa desde entonces fue producir Zope y productos relacionados con él, y ofrecer servicios de adaptación y mantenimiento para todos ellos. Zope Corporation

ha sabido, además, crear una dinámica comunidad de desarrolladores de software libre alrededor de sus productos, y colaborar activamente con ellos.

5.2.4. Fuente de un producto con limitaciones

Este modelo es similar al anterior, pero incluye medidas para limitar la competencia o maximizar los ingresos. Entre las limitaciones más habituales podemos considerar las siguientes:

- Distribución propietaria durante un tiempo, luego libre

Con o sin promesa de distribución libre posterior, cada nueva versión del producto se vende como software propietario. Pasado un tiempo (normalmente, cuando se empieza a comercializar una nueva versión, también como software propietario), esa versión pasa a distribuirse con una licencia libre. De esta manera, la empresa productora obtiene ingresos de los clientes interesados en disponer lo antes posible de nuevas versiones, y a la vez minimiza la competencia, ya que cualquier empresa que quiera competir usando ese producto sólo podrá hacerlo con la versión libre (disponible sólo cuando ya existe una nueva versión propietaria, supuestamente mejor y más completa).

- Distribución limitada durante un tiempo

En este caso, el software es libre desde que se comienza a distribuir. Pero como nada en una licencia libre obliga a distribuir el programa a quien lo quiera (esto es algo que quien tiene el software puede hacer o no), lo que hace el productor es distribuirlo durante un tiempo sólo a sus clientes, que le pagan por ello (normalmente en forma de contrato de mantenimiento). Al cabo de un tiempo, el productor lo distribuye a cualquiera, por ejemplo poniéndolo en un archivo de acceso público. De esta manera, el productor obtiene ingresos de sus clientes, que perciben esta disposición preferente del software como un valor añadido. Naturalmente, el modelo sólo funciona si los clientes a su vez no hacen público el programa cuando lo reciben. Para cierto tipo de clientes, esto puede no ser habitual.

ANOTACIONES

En general, en estos casos las empresas desarrolladoras obtienen los beneficios mencionados, pero no a coste cero. Debido al retraso con el que el producto está disponible para la comunidad del software libre, es prácticamente imposible que ésta pueda colaborar en su desarrollo, por lo que el productor se beneficiará muy poco de contribuciones externas.

Ejemplo

Algunos casos de empresas que utilizan este modelo de negocio son lo siguientes:

- artofcode LLC

Desde el año 2000, artofcode comercializa Ghostscript en tres versiones (anteriormente lo hacía Alladin Enterprises, con un modelo similar). La versión más actual la distribuye como AFPL Ghostscript, bajo una licencia propietaria (que permite el uso y la distribución no comercial). La siguiente (con un retraso de un año, más o menos) la distribuye como GNU Ghostscript, bajo la GNU GPL. Por ejemplo, en el verano del 2003, la versión AFPL es la 8.11 (liberada el 16 de agosto), mientras que la versión GNU es la 7.07 (distribuida como tal el 17 de mayo, pero cuya versión AFPL equivalente es del 2002). Además, artofcode ofrece una tercera versión, con una licencia propietaria que permite la integración en productos no compatibles con la GNU GPL (en este caso usa un modelo dual, que será descrito más adelante).

- Ada Core Technologies

Fue fundada en 1994 por los autores del primer compilador de Ada 95, cuyo desarrollo fue financiado en parte por el Gobierno de EE.UU., y que estaba basado en GCC, el compilador de GNU. Desde el principio sus productos han sido software libre. Pero la mayoría de ellos los ofrecen primero a sus clientes, como parte de un contrato de mantenimiento. Por ejemplo, su compilador, que sigue estando basado en GCC y se distribu-

ye bajo la GNU GPL, se ofrece a sus clientes como Gnat Pro. Ada Core Technologies no ofrece este compilador al público en general de ninguna manera, y normalmente no se encuentran versiones del mismo en la Red. Sin embargo, con un retraso variable (en torno a un año), Ada Core Technologies ofrece las versiones *públicas* de su compilador, muy similares pero sin ningún tipo de soporte, en un archivo de ftp anónimo.

5.2.5. Licencias especiales

En estos modelos, la empresa produce un producto que distribuye bajo dos licencias o más. Al menos una de ellas es de software libre, pero las otras son típicamente propietarias y le permiten vender el producto de una forma más o menos tradicional. Normalmente, estas ventas se complementan con la oferta de consultoría y desarrollos relacionados con el producto. Por ejemplo, una empresa puede distribuir un producto como software libre bajo la GNU GPL, pero ofrecer también una versión propietaria (simultáneamente, y sin retraso para ninguna de las dos) para quien no quiera las condiciones de la GPL, por ejemplo, porque quiere integrar el producto con uno propietario (algo que la GPL no permite).

Ejemplo

Sleepycat Software fue fundada en 1996, y anuncia que desde ese momento ha tenido beneficios (lo que desde luego es notable en una empresa relacionada con el software). Sus productos, incluyendo Berkeley DB (un gestor de datos muy popular y que puede empotrarse fácilmente en otras aplicaciones) se distribuyen bajo una licencia libre que especifica que en caso de empotrarse en otro producto, ha de ofrecerse el código fuente de ambos. Sleepycat ofrece servicios de consultoría y desarrollo para sus productos, pero además los ofrece bajo licencias que permiten empotrarlos sin tener que distribuir el código fuente. Naturalmente, esto lo hace bajo contrato específico, y en general en régimen de venta como software propietario.

5.2.6. Venta de marca

Aunque puedan conseguirse productos muy similares por menos dinero, muchos clientes están dispuestos a pagar el extra por comprar una *marca*. Este principio es utilizado por empresas que invierten en establecer una marca con buena imagen, y bien reconocida, que más adelante les permita vender con suficiente margen productos libres. En muchos casos no sólo venden esos productos, sino que los acompañan de servicios que los clientes aceptarán también como valor añadido.

Los casos más conocidos de este modelo de negocio son las empresas que comercializan distribuciones GNU/Linux. Estas empresas tratan de vender algo que en general se puede obtener a un coste bastante menor en la Red (o en otras fuentes con menos imagen de marca). Por ello han de conseguir que el consumidor reconozca su marca, y esté dispuesto a pagar el sobreprecio. Para ello, no sólo invierten en publicidad, sino que también ofrecen ventajas objetivas (por ejemplo, una distribución bien conjuntada o un canal de distribución que llegue hasta las cercanías del cliente). Además, suelen ofrecer a su alrededor una gran cantidad de servicios, tratando de rentabilizar el máximo posible esa imagen de marca (desde formación hasta programas de certificación para terceras partes).

Ejemplo

La distribución Red Hat Linux comenzó a distribuirse en 1994 (la empresa empezó a conocerse con el nombre actual en 1995). Desde entonces, Red Hat ha conseguido posicionar su nombre como el de *la* distribución de GNU/Linux por excelencia. Hoy día, en torno a ese nombre Red Hat comercializa todo tipo de servicios relacionados con la distribución, con GNU/Linux, y con software libre en general.

5.3. Otras clasificaciones de modelos de negocio

Hay otras clasificaciones de modelos de negocio clásicas en la literatura sobre software libre. A modo de ejemplo ofrecemos a continuación una de las más clásicas.

5.3.1. Clasificación de Hecker

La clasificación que se ofrece en Frank Hecker (mayo, 1998). *Setting up shop: The business of open-source software* fue la más usada por la publicidad de la Open Source Initiative, y también una de las primeras en tratar de categorizar los negocios que estaban surgiendo por aquella época. Sin embargo, incluye varios modelos poco centrados en software libre (en ellos es poco más que un acompañante del modelo principal). En cualquier caso, los modelos que describe son los siguientes:

- *Support seller* (venta de servicios relacionados con el producto).

La empresa promueve un producto libre (que ha desarrollado o en cuyo desarrollo participa activamente) y vende servicios, como consultoría o adaptación a necesidades concretas para él.

- *Loss leader* (venta de otros productos propietarios).

En este caso, el programa libre se utiliza para promover de alguna forma la venta de otros productos propietarios relacionados con él.

- *Widget frosting* (venta de hardware).

El negocio fundamental es la venta de hardware, y el software libre se considera un complemento para él, que puede ayudar a la empresa a obtener una ventaja competitiva.

- *Accessorizing* (venta de accesorios).

Se comercializan productos relacionados con el software libre, como libros, dispositivos informáticos, etc.

- *Service enabler* (venta de servicios).

El software libre sirve para crear un servicio (normalmente accesible en línea) del que la empresa obtiene algún beneficio.

- *Brand licensing* (venta de marca).

Una empresa registra marcas que consigue asociar con programas libres, probablemente desarrollados por ella. Luego obtiene ingresos cuando vende derechos del uso de esas marcas.

- *Sell it, free it* (vende, libera).

Modelo similar a *loss leader*, pero realizado de forma cíclica. Primero se comercializa un producto como software libre. Si se consigue que tenga cierto éxito, la siguiente versión se distribuye como software propietario durante un tiempo, y pasado éste también se libera. Para entonces, se empieza a distribuir una nueva versión propietaria, y así sucesivamente.

- *Software franchising* (franquicia de software).

Una empresa franquicia el uso de sus marcas, relacionadas con un programa libre determinado.

Nota

El lector habrá podido observar cómo esta clasificación es bastante diferente de la que hemos ofrecido nosotros, pero aun así algunas de sus categorías coinciden casi exactamente con algunas de las nuestras.

5.4. Impacto sobre las situaciones de monopolio

El mercado informático tiende a la dominación de un producto en cada uno de sus segmentos. Los usuarios quieren rentabilizar el esfuerzo realizado en aprender cómo funciona un programa, las empresas quieren encontrar gente formada en el uso de su software, y todos quieren que los datos que gestionan puedan ser entendidos por los programas de las empresas y por las personas con las que se relacionan. Por esta razón, cualquier iniciativa dedicada a romper una situación *de facto* donde un producto domina claramente el mercado está destinada a producir más de lo mismo: si tiene éxito,