ieu
ONLINE

Semana 3
Software Libre

Lectura
How I coined the term ‘open source’

Peterson, C. (1 de Febrero de 2018). How I coined the term ‘open source’. Recuperado de: https://opensource.com/article/18/2/coining-term-open-source-software

How I coined the term 'open source'

Christine Peterson. (2018). OpenSource.com
Recuperado de: https://opensource.com/article/18/2/coining-term-open-source-software

In a few days, on February 3, the 20th anniversary of the introduction of the term "open source software" is upon us. As open source software grows in popularity and powers some of the most robust and important innovations of our time, we reflect on its rise to prominence.

I am the originator of the term "open source software" and came up with it while executive director at Foresight Institute. Not a software developer like the rest, I thank Linux programmer Todd Anderson for supporting the term and proposing it to the group.

This is my account of how I came up with it, how it was proposed, and the subsequent reactions. Of course, there are a number of accounts of the coining of the term, for example by Eric Raymond and Richard Stallman, yet this is mine, written on January 2, 2006.

It has never been published, until today.

The introduction of the term "open source software" was a deliberate effort to make this field of endeavor more understandable to newcomers and to business, which was viewed as necessary to its spread to a broader community of users. The problem with the main earlier label, "free software," was not its political connotations, but that—to newcomers—its seeming focus on price is distracting. A term was needed that focuses on the key issue of source code and that does not immediately confuse those new to the concept. The first term that came along at the right time and fulfilled these requirements was rapidly adopted: open source.

This term had long been used in an "intelligence" (i.e., spying) context, but to my knowledge, use of the term with respect to software prior to 1998 has not been confirmed. The account below describes how the term open source software caught on and became the name of both an industry and a movement.

Meetings on computer security

In late 1997, weekly meetings were being held at Foresight Institute to discuss computer security. Foresight is a nonprofit think tank focused on nanotechnology and artificial intelligence, and software security is regarded as central to the reliability and security of both. We had identified free software as a promising approach to improving software security and reliability and were looking for ways to promote it. Interest in free software was starting to grow outside the programming community, and it was increasingly clear that an opportunity was coming to change the world. However, just how to do this was unclear, and we were groping for strategies.

At these meetings, we discussed the need for a new term due to the confusion factor. The argument was as follows: those new to the term "free software" assume it is referring to the price. Oldtimers must then launch into an explanation, usually given as follows: "We mean free as in freedom, not free as in beer." At this point, a discussion on software has turned into one about the price of an alcoholic beverage. The problem was not that explaining the meaning is impossible—the problem was that the name for an important idea should not be so confusing to newcomers. A clearer term was needed. No political issues were raised regarding the free software term; the issue was its lack of clarity to those new to the concept.

Releasing Netscape

On February 2, 1998, Eric Raymond arrived on a visit to work with Netscape on the plan to release the browser code under a free-software-style license. We held a meeting that night at Foresight's office in Los Altos to strategize and refine our message. In addition to Eric and me, active participants included Brian Behlendorf, Michael Tiemann, Todd Anderson, Mark S. Miller, and Ka-Ping Yee. But at that meeting, the field was still described as free software or, by Brian, "source code available" software.

While in town, Eric used Foresight as a base of operations. At one point during his visit, he was called to the phone to talk with a couple of Netscape legal and/or marketing staff. When he was finished, I asked to be put on the phone with them—one man and one woman, perhaps Mitchell Baker—so I could bring up the need for a new term. They agreed in principle immediately, but no specific term was agreed upon.

Between meetings that week, I was still focused on the need for a better name and came up with the term "open source software." While not ideal, it struck me as good enough. I ran it by at least four others: Eric Drexler, Mark Miller, and Todd Anderson liked it, while a friend in marketing and public relations felt the term "open" had been overused and abused and believed we could do better. He was right in theory; however, I didn't have a better idea, so I thought I would try to go ahead and introduce it. In hindsight, I should have simply proposed it to Eric Raymond, but I didn't know him well at the time, so I took an indirect strategy instead.

Todd had agreed strongly about the need for a new term and offered to assist in getting the term introduced. This was helpful because, as a non-programmer, my influence within the free software community was weak. My work in nanotechnology education at Foresight was a plus, but not enough for me to be taken very seriously on free software questions. As a Linux programmer, Todd would be listened to more closely.

The key meeting

Later that week, on February 5, 1998, a group was assembled at VA Research to brainstorm on strategy. Attending—in addition to Eric Raymond, Todd, and me—were Larry Augustin, Sam Ockman, and attending by phone, Jon "maddog" Hall.

The primary topic was promotion strategy, especially which companies to approach. I said little, but was looking for an opportunity to introduce the proposed term. I felt that it wouldn't work for me to just blurt out, "All you technical people should start using my new term." Most of those attending didn't know me, and for all I knew, they might not even agree that a new term was greatly needed, or even somewhat desirable.

Fortunately, Todd was on the ball. Instead of making an assertion that the community should use this specific new term, he did something less directive—a smart thing to do with this community of strong-willed individuals. He simply used the term in a sentence on another topic—just dropped it into the conversation to see what happened. I went on alert, hoping for a response, but there was none at first. The discussion continued on the original topic. It seemed only he and I had noticed the usage.

Not so—memetic evolution was in action. A few minutes later, one of the others used the term, evidently without noticing, still discussing a topic other than terminology. Todd and I looked at each other out of the corners of our eyes to check: yes, we had both noticed what happened. I was excited—it might work! But I kept quiet: I still had low status in this group. Probably some were wondering why Eric had invited me at all.

Toward the end of the meeting, the question of terminology was brought up explicitly, probably by Todd or Eric. Maddog mentioned "freely distributable" as an earlier term, and "cooperatively developed" as a newer term. Eric listed "free software," "open source," and "sourceware" as the main options. Todd advocated the "open source" model, and Eric endorsed this. I didn't say much, letting Todd and Eric pull the (loose, informal) consensus together around the open source name. It was clear that to most of those at the meeting, the name change was not the most important thing discussed there; a relatively minor issue. Only about 10% of my notes from this meeting are on the terminology question.

But I was elated. These were some key leaders in the community, and they liked the new name, or at least didn't object. This was a very good sign. There was probably not much more I could do to help; Eric Raymond was far better positioned to spread the new meme, and he did. Bruce Perens signed on to the effort immediately, helping set up Opensource.org and playing a key role in spreading the new term.

For the name to succeed, it was necessary, or at least highly desirable, that Tim O'Reilly agree and actively use it in his many projects on behalf of the community. Also helpful would be use of the term in the upcoming official release of the Netscape Navigator code. By late February, both O'Reilly & Associates and Netscape had started to use the term.

Getting the name out

After this, there was a period during which the term was promoted by Eric Raymond to the media, by Tim O'Reilly to business, and by both to the programming community. It seemed to spread very quickly.

On April 7, 1998, Tim O'Reilly held a meeting of key leaders in the field. Announced in advance as the first "Freeware Summit," by April 14 it was referred to as the first "Open Source Summit."

These months were extremely exciting for open source. Every week, it seemed, a new company announced plans to participate. Reading Slashdot became a necessity, even for those like me who were only peripherally involved. I strongly believe that the new term was helpful in enabling this rapid spread into business, which then enabled wider use by the public.

A quick Google search indicates that "open source" appears more often than "free software," but there still is substantial use of the free software term, which remains useful and should be included when communicating with audiences who prefer it.

A happy twinge

When an early account of the terminology change written by Eric Raymond was posted on the Open Source Initiative website, I was listed as being at the VA brainstorming meeting, but not as the originator of the term. This was my own fault; I had neglected to tell Eric the details. My impulse was to let it pass and stay in the background, but Todd felt otherwise. He suggested to me that one day I would be glad to be known as the person who coined the name "open source software." He explained the situation to Eric, who promptly updated his site.

Coming up with a phrase is a small contribution, but I admit to being grateful to those who remember to credit me with it. Every time I hear it, which is very often now, it gives me a little happy twinge.

The big credit for persuading the community goes to Eric Raymond and Tim O'Reilly, who made it happen. Thanks to them for crediting me, and to Todd Anderson for his role throughout. The above is not a complete account of open source history; apologies to the many key players whose names do not appear. Those seeking a more complete account should refer to the links in this article and elsewhere on the net.

ieu
ONLINE

Semana 3
Software Libre

Lectura
Historia y desarrollo del software libre y de código fuente

Catalunya, U. O. (2005). Historia y desarrollo del software libre y de código fuente. Catalunya: Universitat Oberta de Catalunya.

1) HISTORIA Y DESARROLLO DEL SOFTWARE LIBRE Y DE CÓDIGO FUENTE ABIERTO

El *open source* es, en primer lugar, un tipo de organización social de la producción. Se originó paralelamente al desarrollo de software, y primordialmente se refiere al acceso abierto al código fuente de un programa o aplicación. El código fuente es un conjunto de instrucciones, una lista de órdenes y de pautas que constituye la fórmula fundamental de un paquete de software. Habitualmente se utiliza el símil del ADN para referirse a él.

La mayoría de software comercial está escrito en lenguaje binario, un código que las máquinas pueden entender pero que resulta ilegible para los humanos. El código fuente es la fórmula que origina este lenguaje binario, con éste es posible comprender la lógica de un programa. Así, cualquiera que tenga conocimientos técnicos puede formular propuestas de modificación del mismo. El código fuente se articula alrededor de una noción "especial" de propiedad. La mayoría de software comercial está basado en el control de los derechos de propiedad del código fuente. Los distintos programas informáticos son vendidos a los usuarios, viéndose éstos obligados a aceptar este producto final tal y como se les entrega porque al no poder acceder a la lógica del programa (código fuente), resulta imposible cualquier tipo de modificación.

Convencionalmente, en una economía capitalista, la propiedad es el derecho de exclusión de los "otros" respecto del uso de un bien o servicio. En *open source* la propiedad se configura fundamentalmente partiendo del derecho de distribución y de no exclusión. El código fuente se publica y se distribuye, pudiendo ser modificado por todos aquellos que lo deseen. Gracias al hecho que el código fuente se conoce, los usuarios pueden acceder a él y, si lo consideran necesario, modificarlo o incluso generar nuevas aplicaciones. En *open source* el código fuente es libre, abierto, público y no propietario. Esta nueva forma de propiedad, que es completamente contradictoria con el régimen habitual y tradicional de derechos de propiedad intelectual, se apoya en una lógica de trabajo basada en la motivación humana.

Contrariamente a lo que algunos puedan pensar, el *open source* no es una fantasía o una práctica marginal. Existen proyectos de desarrollo extensos y consolidados que son fruto de un proceso de producción enteramente "en abierto". Los más populares son Linux y Apache, pero hay muchos más. Lo más interesante del fenómeno es que se trata de una práctica en expansión en el mundo de la investigación, los *hackers*, la educación… y también en la esfera institucional y de los negocios, incluyendo algunas grandes corporaciones, como IBM.

El modelo *open source* introduce, como hemos mencionado antes, un nuevo tipo de producción, cooperativo, que trasciende los límites tradicionales de la división social del trabajo, propios de las hierarquías. Trabajar en *open source* implica trabajar en una red abierta de cooperación voluntaria. Aunque el *open source* no empezó con Internet, la Red se ha consolidado como la plataforma fundamental de la escalabilidad y la interactividad del proceso de cooperación.

El modelo *open source* es compatible con distintas lógicas sociales y valores, y no es necesariamente anti-capitalista puesto que hay muchas empresas capitalistas, incluyendo grandes corporaciones, que siguen este modelo de trabajo. Sin embargo, sí que se trata de un modelo a-capitalista. No necesita el incentivo de los beneficios para trabajar y no se basa en la apropiación privada del derecho exclusivo para usar y disfrutar de un producto. Está basado en un tipo de organización social que tiene profundas implicaciones políticas y puede afectar nuestra percepción de la necesidad de preservar las instituciones capitalistas y las hierarquías de producción para manejar las necesidades de un mundo complejo.

El contexto del *open source*

El contexto que enmarca el desarrollo del *open source*, entendido como un fenómeno social, un fenómeno político, y un fenómeno económico, incluye como mínimo cuatro grandes características:

1) Internet transforma la naturaleza del proceso productivo, encadenando interactividad y distribución. La organización en red deviene efectiva, particularmente con el incremento del ancho de banda de las telecomunicaciones. En el paradigma de producción *open source* tan importante como el código en sí lo es el proceso mediante el cual se ha generado.

2) *Open source* expresa el desarrollo de nuevas relaciones entre la comunidad, la cultura y la actividad comercial. La comunidad *open source* está basada en un conjunto de normas y valores comunes. Además, partiendo de esta autonomía cultural, la comunidad se relaciona con las reglas de la organización capitalista que gobiernan un contexto más amplio. De hecho, en la historia de la organización industrial, las ideas crean instituciones que erigen procesos de producción. Así, las ideas intrínsecas en el *open source* están en las raíces de una nueva lógica de producción.

3) El *open source* fundamenta la nueva lógica de la organización de la producción en un procedimiento económico de conocimiento intensivo. El desarrollo de software se hace mediante conocimiento codificado que se combina desde la base en el proceso de producción. Así, como hemos mencionado anteriormente, *open source* es un "experimento" de producción construido alrededor de una noción distinta de propiedad. La concepción tradicional de propiedad se basa en el derecho a excluir los no propietarios del uso de algo que es propiedad de un tercero. Igualmente, la propiedad *open source*, está configurada en torno al derecho de distribución, no el derecho de exclusión. Esta particularidad ya la encontramos en la tradición del "fair use" de creaciones intelectuales que son usadas sin securizar su propiedad. Bajo una noción extendida del "fair use" ningún uso individual de éste será permitido para constreñir el subsiguiente uso por parte de otra persona y con cualquier otro propósito.[1]

[1] Para un análisis exhaustivo de la transformación de la noción de "propiedad intelectual" y el "fair use", consultar la obra de Lawrence Lessig, *Free culture. How big media uses technology and the law to lock down culture and control creativity*. (2004)

4) El fenómeno *open source* es intrínsecamente social, no se limita al campo del software, sino que es aplicable al conjunto de producción y distribución de conocimiento.

Historia del movimiento / práctica *open source*

El comienzo de la industria digital está marcado por la ausencia de distinción entre hardware y software. Los productores y los usuarios de cada máquina tenían que escribir el software necesario para poder interactuar con ella y, consecuentemente, este lenguaje *ad hoc* no podía ser usado por otras máquinas. En 1956 el Departamento de Justicia de los Estados Unidos, bajo la ley anti-trust, decidió que las empresas Western Electric y ATT (American Telephone and Telegraph) no podían unirse para manufacturar otros productos que no fueran telecomunicaciones. Los abogados de ATT, en una estrategia preventiva, promovieron las licencias de software y otras tecnologías de comunicación a una tarifa nominal, evitando así ser acusados de producir software (un producto ajeno a las telecomunicaciones) para beneficio propio. Esta decisión permitió transferir al dominio público las investigaciones en materia de software que se habían llevado a cabo en los Laboratorios Bell.

En 1964, investigadores del MIT (Massachussets Institute of Technology), en cooperación con los Laboratorios Bell y General Electric, desarrollaron un sistema operativo de tiempo compartido llamado MULTICS (Multiplexed Information and Computing Service). El componente de software del proyecto era difícil de implementar y en 1969 los Laboratorios Bell se retiraron. Sin embargo dos investigadores de Bell, Ken Thompson y Dennis Ritchie, decidieron continuar por su cuenta. En verano de 1969 crearon el kernel de un sistema operativo al que llamaron UNICS (un chiste para mofarse de MULTICS), aunque posteriormente pasó a denominarse UNIX. Actualmente UNIX es el principal sistema operativo de Internet, así como el más utilizado por investigadores y la base de grandes paquetes de software.

En octubre de 1973 Thompson y Ritchie presentaron un "paper" sobre UNIX en el simposio de ACM (Association for Computing Machinery) y algunas de las comunidades de

programadores más innovadoras empezaron a mostrar interés. Algunas limitaciones legales llevaron a ATT-Bell a licenciar UNIX, en primer lugar para universidades e investigadores, y posteriormente para organizaciones comerciales y militares.

El software era cedido sin soporte técnico, de modo que los usuarios empezaron a mejorar el programa por sí mismos. ATT facilitaba, a cambio de unos cientos de dólares, el código fuente escrito en el lenguaje de programación C. UNIX funcionaba en cualquier máquina que tuviera un compilador de tipo C, no sólo en máquinas DEC en las cuales se había desarrollado. Así mismo, UNIX proporcionaba un instrumento docente que podía ser comprendido y modificado gracias a la disponibilidad del código fuente y rápidamente se convirtió en una herramienta muy popular en los departamentos de informática. Su difusión fue muy rápida y pronto se expandió por todo el mundo. El interés por Internet todavía era muy limitado y la comunicación de los resultados por parte de los desarrolladores de UNIX fundamentalmente tenía lugar en encuentros y seminarios que, previsiblemente, derivaron en una comunidad internacional de usuarios de UNIX.

Uno de los primeros departamentos de informática que adoptó UNIX fue el de la Universidad de Berkeley. En otoño de 1975 dos profesores, Michael Stonebraker y Robert Fabry, consiguieron comprar un ordenador nuevo y potente, un PDP-11/70, para trabajar con UNIX. Simultáneamente, Ken Thompson llegó a Berkeley para pasar allí un año sabático. Un nuevo grupo de estudiantes graduados se sumó al departamento, incluyendo Bill Joy y Chuck Haley. Con la ayuda de Thompson, en verano de 1976, Joy y Haley, trabajando con el sistema Pascal, fueron capaces de mejorar el kernel de UNIX. La noticia pronto se difundió entre la comunidad de usuarios de UNIX, y Joy elaboró un paquete de herramientas y utilidades al que llamó Berkeley Software Distribution (BSD). A partir de ese momento, BSD se fue perfeccionando y se generaron distintas versiones. Esto fue posible gracias a la colaboración de los estudiantes de Berkeley y de otros programadores de UNIX distribuidos por el mundo, incluyendo un grupo de adolescentes de un instituto de Boston, quienes introdujeron grandes mejoras en la versión 7 de BSD, permitiendo que el programa fuera migrado a distintas máquinas.

Dada la gran demanda existente de UNIX, la ATT empezó a pensar estrategias para controlar el producto. Se restringió el uso de su licencia a una sola universidad (Berkeley) y únicamente para usos vinculados a docencia e investigación. A finales de 1979, Joy liberó 3BSD, una distribución completa del sistema operativo UNIX sobre un potente procesador VAX. Este fue el factor más importante que tuvo en cuenta la DARPA/ARPA[2] (agencia de investigación del Pentágono, y espónsor de ARPANET) cuando decidió utilizar UNIX en sus redes de ordenadores.

La ARPA financió al profesor Fabry la creación de un grupo de investigación en Berkeley, el Computer Systems Research Group (CSRG), que liberó en 1980 la versión 4BSD y en 1983 la versión 4.2BSD, un sistema capaz de integrar "TCP/IP networking" (protocolos de comunicación de Internet) en UNIX. 4.2BSD es el software base de Internet tal y como la conocemos hoy. Paralelamente, en 1984 la ATT y los Laboratorios Bell se separaron por orden judicial, permitiéndose así que Bell comercializara sus investigaciones. Una de las primeras acciones que llevó a cabo fue la creación de un laboratorio de desarrollo del sistema UNIX. Se empezó a licenciar al precio de cientos de miles de dólares y a partir de entonces, BSD y ATT-UNIX emprendieron caminos distintos. La ATT inició una serie de litigaciones para restringir la difusión libre de BSD y de UNIX en general.

Como reacción a este intento de los Laboratorios Bell de comercializar y restringir el uso de UNIX, un grupo de programadores del laboratorio de inteligencia artificial del MIT, liderado por Richard Stallman, creó en 1984 la Free Software Foundation. El objetivo fundamental era la construcción de un sistema operativo que, aunque basado en la tradición UNIX, fuera distribuido y usado libremente; lo llamaron GNU (GNU is Not UNIX). Stallman publicó el mismo año el Manifiesto GNU en el que se recogió una definición del término “libre”, vinculándolo a “free speech” en lugar de “free beer”, distinguiendo así entre libre y gratis. El manifiesto recoge cuatro libertades básicas basadas en el acceso libre y total al código fuente de los programas:

[2] El nombre de la agencia ha ido cambiando a lo largo de los años. Las denominaciones ARPA (Advanced Research Projects Agency) y DARPA (Defense Advanced Research Projects Agency) se pueden usar indistintamente.

- Libertad 0: libertad para utilizar un programa, sea cual sea el propósito.
- Libertad 1: libertad para estudiar cómo funciona un programa, y capacidad para adaptarlo a las propias necesidades. El acceso al código fuente es una condición *sine qua non*.
- Libertad 2: libertad para redistribuir copias.
- Libertad 3: libertad para mejorar un programa y presentar dichas mejoras a la comunidad para que pueda beneficiarse. Aquí el acceso al código fuente también es una condición *sine qua non*.

Stallman creó una herramienta legal e institucional para reforzar estas libertades, la General Public License (GPL), que substituye el término *copyright* por *copyleft*. El software licenciado con GPL, así como los productos que se deriven de éste, no pueden convertirse en productos propietarios (privativos siguiendo la terminología de Stallman). Además, los códigos de programas GPL no pueden ser usados en ninguna combinación con software propietario a no ser que el nuevo software se licencie también con GPL (cláusula viral). El pasado 22 y 23 de junio se celebró en Barcelona la tercera conferencia internacional sobre la GPLv3. Organizada por la delegación europea de la Free Software Foundation, el objetivo fundamental de este encuentro es contribuir a la difusión de esta licencia así como realizar aportaciones a la redacción del borrador de esta tercera versión.

La Free Software Foundation ha generado software de calidad, particularmente el compilador GCC y el editor de textos *GNU Emacs*. Sin embargo los recursos de la FSF eran limitados y nunca fueron capaces de construir un sistema operativo entero bajo GPL. Así, la GPL y la cultura implícita en la FSF se convirtieron en elementos más importantes que el propio software producido por Stallman y sus colaboradores. Esto fue así particularmente porque Stallman trabajaba con pequeños grupos de programadores, sometidos a un estricto control intelectual, y porque omitieron el poder cooperativo que propiciaba Internet. Así, a finales de 1980, GPL/GNU fue un movimiento limitado y restringido a pequeñas redes de programadores.

Paralelamente, BSD contaba con un grupo bastante extenso de aplicaciones comerciales (la más notable era de Sun Microsystems). No obstante, el grupo de investigación

original de Berkeley (CRSG) se vio inmerso en una serie de litigaciones con ATT y la Universidad de California, siendo finalmente disuelto en 1995. En el contexto corporativo, varios usuarios potentes de UNIX (incluyendo entre otros IBM, HP, Siemens…) crearon una Fundación Open Source para contradecir el intento de ATT de controlar UNIX, pero la crisis que afectó a la industria informática en 1991 llevó a la disolución de esta fundación.

A principios de los años 90 parecía que la idea de software de código fuente abierto estaba limitada a programadores contraculturales que no tenían las herramientas necesarias para desarrollar sus sueños y sus proyectos. Microsoft estuvo al borde de lograr el monopolio en el campo del software de sistemas operativos, y UNIX y sus derivados estaban siendo totalmente privatizados. En ese momento, dos grandes sucesos invirtieron la tendencia: Linux y la expansión de Internet.

Linux nació fruto de las necesidades de Linus Torvalds, un estudiante de informática de 21 años de la Universidad de Helsinki. En 1991, mientras cursaba un master, precisó de UNIX pero al no tener acceso a un ordenador potente quiso adaptar el sistema a su nuevo PC 386. Sintió que necesitaría ayuda y empezó escribiendo un kernel muy primitivo para su programa (el kernel de un sistema operativo es el compendio de instrucciones que controla el proceso de información y las asignaciones de recursos que hacen que un ordenador funcione). En otoño del mismo año, Torvalds liberó la versión 0.02 de Linux (la llamó Freix, pero el administrador de sistemas que colgó el programa la rebautizó como Linux). Torvalds puso su kernel en Internet y solicitó ayuda para finalizar su desarrollo. Naturalmente, estaba a disposición de la comunidad el código fuente y se autorizaba explícitamente su modificación, esperando que los hipotéticos cambios fueran también liberados de modo que aquellos interesados en el proyecto que trabajaban en red pudieran beneficiarse del trabajo común. Visto el creciente interés, en enero de 1992 Torvalds liberó una nueva versión (Linux 0.12) bajo licencia GPL, incluyendo la "cláusula viral", de modo que se garantizaba que las contribuciones al desarrollo de Linux serían abiertas y libres; de hecho deberíamos utilizar el término GNU/Linux puesto que Torvalds aportó el kernel y el conjunto de aplicaciones complementarias surgieron de la FSF.

Linux se desarrolló, desde los inicios, mediante una red de cooperación que debatía abiertamente en Internet, y que, lejos de caer en disputas ideológicas, centró el debate en cuestiones meramente técnicas. Habiendo nacido libre, nunca surgieron litigios con abogados de empresas de software (sólo en 2002 empezaron algunas maniobras por parte de Microsoft, pretendiendo que Linux se había apropiado de software propiedad de otras compañías, campañas de escaso éxito en los tribunales y que se alargaron hasta 2004). Linux creció rápidamente y derivó en una comunidad de miles de colaboradores, con un núcleo de varios cientos de programadores, y una cifra estimada de 18 millones de usuarios en 2001, y 29 millones en 2005[3]. Aunque el proyecto se expandió y la comunidad se multiplicó, Linux consiguió evitar el *forking* (desarrollo por parte de un segmento de la comunidad de una versión distinta a partir del código fuente matriz) evitando así la división de los esfuerzos colaborativos, puesto que las distintas versiones tenían garantizada la compatibilidad entre ellas. Este hecho fue consecuencia mayoritariamente de la predisposición de Torvalds a aceptar modificaciones substanciales del código, y también al hecho que se permitía a otras personas formular propuestas acerca de éste así como controlar aspectos clave del mismo. Un ejemplo concreto de esta filosofía de trabajo fue el desarrollo de un sistema operativo específico para Internet sin usar código vinculado a las disputas entre ATT y Berkeley por BSD. El esfuerzo del desarrollador inglés Alan Cox, fue ampliamente reconocido por Linus Torvalds y se integró en el programa.

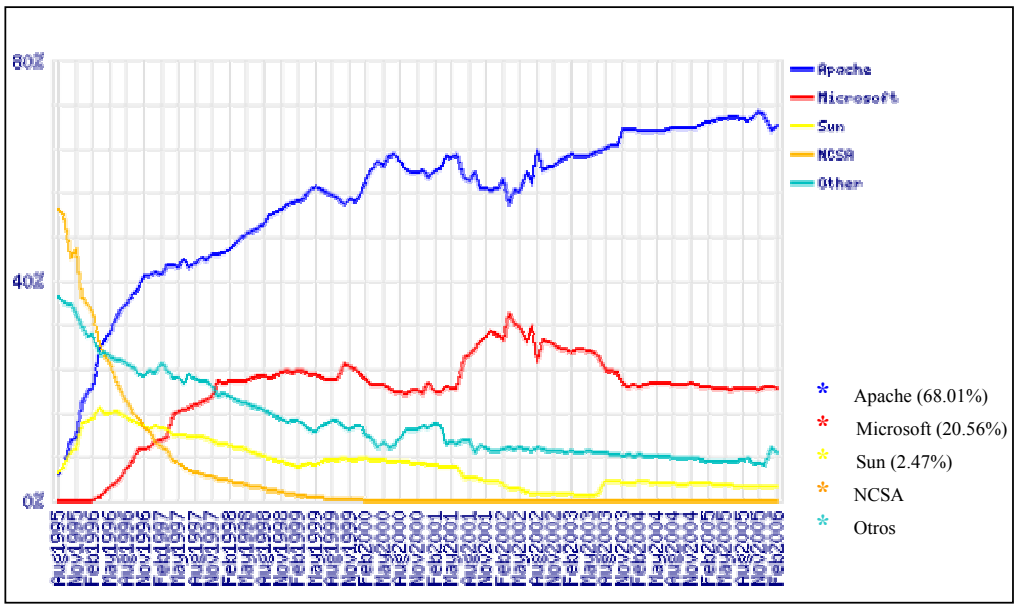
Como veremos más adelante, Linux ha sido aceptado gradualmente por grandes corporaciones, gobiernos e instituciones, así como por un creciente número de negocios que desarrollan aplicaciones y servicios para este sistema operativo. Esto sin renunciar a su carácter de programa *open source* completo, sin apropiaciones privadas y sin obtención de beneficios por parte de Linus Torvalds o alguno de los desarrolladores principales del proyecto. Es más, Linux continua siendo desarrollado y mantenido por una comunidad global que interactúa en Internet sin ninguna hierarquía formal.

El otro gran experimento en *open source* al que nos referíamos anteriormente es el programa Apache Web Server (figura 1). Un software web server (servidor web) es el

[3]FUENTE: The Linux Counter (http://counter.li.org/) [Fecha de consulta: 20 de julio de 2005]

conjunto de aplicaciones que responden a la demanda de páginas web efectuada por los ordenadores. El primer web server fue *Daemon*, desarrollado por Rob Mc Cool en la Universidad de Illinois (National Center for Supercomputing Applications). Cuando Mc Cool dejó el centro en 1994, la administración no prosiguió con su trabajo hasta que fue retomado en 1995 por Brian Behlendorf, un estudiante de Berkeley que trabajaba en la versión digital de la revista *Wired* y era miembro de Cyborganics (una comunidad contracultural físico/virtual de San Francisco). Gracias al hecho que el código del NCSA era *open source*, Behlendorf pudo retomar el proyecto. Junto a ocho desarrolladores más empezó el trabajo y tres meses después ya contaban con 150 subscriptores a la lista de correo del proyecto. En diciembre de 1995 liberaron Apache 1.0. Muchos de los desarrolladores implicados estaban vinculados paralelamente a actividades de desarrollo comercial de web sites, y finalmente liberaron el código bajo licencia BSD en lugar de GPL. La diferencia fundamental es que BSD requiere la liberación del código que se ha generado a partir de contribuciones *open source* pero no impide la mezcla con software propietario con usos comerciales siempre que se mantenga el acceso libre a la información que fue originalmente liberada en formato abierto (no contiene la "cláusula viral").

Fig. 1 - Evolución trimestral de los usos de servidores de Internet (agosto 1995 - febrero 2006)[4]



[4] Fuente: www.netcraft.com [Fecha de consulta: 24 de febrero de 2006]

El grupo Apache está organizado según una red fija de contribuidores. Consta de un comité electo rotatorio, y está dotado de procedimientos de votación y elección, asumiendo pues una especie de constitución interna. Apache ha seguido desarrollando y mejorando sus productos siempre como un proyecto *open source*. Los datos son, como se puede apreciar en el gráfico anterior, reveladores. Actualmente es el programa que utilizan más de dos tercios de los servidores de Internet, además de ser el espolón de un número considerable de negocios, incluyendo la mayoría de operaciones de IBM vinculadas a Internet.

El paso de software libre a *open source*

Linux 2.0 fue liberado en Junio de 1996 y no sólo representó un paso definitivo en el apartado tecnológico (madurez del proyecto) sino que también significó un punto de partida respecto del anclaje ideológico expresado por la FSF. La comunidad Linux no estaba interesada en erradicar el capitalismo o en cambiar los derechos convencionales de propiedad. El objetivo común era desarrollar buen software, y asegurarse que las condiciones de acceso libre al código fuente serían respectadas, porque esa era la clave para producir buenos programas. Además, un número creciente de gente con orientación al mundo de los negocios, como por ejemplo Tom O'Reilly, estaba intentando hacer compatible la libertad del conocimiento con las aplicaciones comerciales de Linux y otros programas creados por la comunidad libre de programadores. Eric Raymond, portavoz intelectual del movimiento Free Software, autor del manifiesto *de facto* de la comunidad (publicado en 1999 en el libro *The Cathedral and the Bazaar*) también apoyaba esta estrategia. En una reunión organizada en las oficinas de VA Linux (actualmente VA Software) en febrero de 1998, Raymond, O'Reilly y los responsables de VA Linux y Red Hat propusieron el término *open source*, que fue aprobado por la cumbre de software libre en abril del mismo año. La nueva definición se basaba en la GPL pero también incorporaba otras formas de licencia, inspiradas por la práctica de Debian. La BSD (Berkeley Software Distribution) también se podía acomodar bajo esta nueva definición.

La definición del término *open source*, consensuada como resultado final de este proceso, contempla que "el programa debe incluir el código fuente y debe permitir su distribución en el código fuente así como de forma compilada"[5]. En el apartado relativo a software derivado, "la licencia debe permitir modificaciones y trabajos derivados, y debe permitir que podrán ser distribuidos bajo los mismos términos que los recogidos en el software original"[6]. Es importante fijarse en el hecho que la definición de *open source* explicita la palabra "permitir", dotándose de un carácter opcional que en la GPL se mantiene como obligatorio. BSD, en cambio, permite la distribución del código fuente abierto pero también mantiene la posibilidad de una distribución cerrada del mismo. Esto significa que los usuarios pueden combinar *free software* y software *open source* según sus necesidades. El compromiso adoptado en estos principios permitió al movimiento *open source* expandirse y convertirse en una práctica común a nivel social y comercial.

Open source y modelos de negocio

Durante un largo periodo de tiempo, el mundo del software evolucionó siguiendo dos líneas incompatibles: software libre desarrollado por comunidades de programadores voluntarios, y software propietario creado por firmas comerciales (con finalidades comerciales o para uso interno). Con la comercialización de distintas versiones de BSD y, posteriormente, con el éxito de Linux, las barreras entre estos dos mundos se fueron desdibujando. Microsoft se aferró a su monopolio que, aunque tecnológicamente era inferior, supo imponer mediante prácticas comerciales y estrategias de negocio cuestionables que en distintas ocasiones terminaron en los tribunales o siendo objeto de elevadas multas. No obstante, una parte substancial del mundo de los negocios basados en tecnologías de la información encontró en el *open source* la oportunidad de desarrollar aplicaciones comerciales. Algunas compañías como VA Linux o Red Hat distribuyeron paquetes Linux respetando las normas *open source* y, lo más sorprendente… ¡ganaron dinero con ello! (beneficios fruto del empaquetado, instrucciones de uso, servicio técnico…) Así, desde 1998 el número de grandes empresas que usaba Linux como la base de sus paquetes de software empezó a crecer, siempre respectando las pautas *open source*. Este fue el caso de Oracle, por

[5] *The program must include source code, and must allow distribution in source code as well as compiled form.*
[6] *The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.*

ejemplo, que empezó a usar Linux combinado con FreeBSD, pero hay otros ejemplos. Computer Associates, Informix, SAP, Hewlett Packard, Dell, Silicon Graphics, Gateway, y quizás el ejemplo que fue más noticiable, IBM. El año 2000 cualquier empresa estaba en condiciones usar Red Hat Linux en un servidor Dell corriendo sobre una base de datos Oracle y recibir asistencia y soporte por parte de IBM.

Simultáneamente, algunas compañías empezaron a liberar el código fuente de su software. Este fue el caso de Netscape con Communicator 5.0 (enero de 1998). El objetivo final era tratar de minimizar el impacto de la competencia ilícita llevada a cabo por Microsoft con su Internet Explorer. Sin embargo, para poder mantener cierto control comercial sobre el producto, Netscape liberó dicho código bajo dos licencias: Netscape Public License (con algunas restricciones) y Mozilla Public License (similar a la GPL). Las restricciones existentes y la ausencia de claridad en la relación entre ambas licencias desencadenó cierta aversión por parte de la comunidad de programadores respecto la posibilidad de cooperar en el desarrollo del código bajo la licencia de Mozilla. Finalmente la iniciativa fue un fracaso.

La estrategia llevada a cabo por IBM para el desarrollo de su software de servidor web fue distinta. Después de rechazar la adquisición de Netscape y una relación de *partnership* con Microsoft, IBM consideró que Apache era la mejor opción y también la más difundida. Pero en lugar de apropiarse de ella o desarrollar un programa propio, algo que estaba en condiciones de hacer, IBM se unió al proyecto (es importante destacar que fue IBM quien se unió al proyecto Apache y no fue al revés). IBM obtuvo un lugar en el comité de Apache y se comprometió a colaborar al desarrollo del programa y a liberar sus contribuciones al código en formato *open source*. Al mismo tiempo, IBM empezó a prestar servicios de apoyo a compañías usando el software de Apache en sus servicios web, algo que ayudó considerablemente al desarrollo de Apache en el mundo empresarial y corporativo. En diciembre de 1998, Lou Gerstner, presidente visionario de IBM, decidió apoyar formalmente el software *open source* así como Linux como parte sustancial de la estrategia de la compañía. Para mostrar su compromiso a la comunidad, IBM liberó su propio código fuente, basado en Linux, para que fuera usado en una de sus mayores computadoras centrales, la System 390.

Microsoft

Microsoft no fue ninguna excepción y no pudo negar lo que era una obviedad. En agosto de 1998 Vinod Valloppillil, un alto ejecutivo de la corporación, escribió un documento confidencial que, una vez difundido, se conoció como "los documentos de Halloween" (*The Halloween documents*). En este memo Valloppillil reconocía la calidad de Linux y la efectividad del modelo *open source* como un método de programación, identificándolo directamente como una amenaza directa a Microsoft. Citando al propio Valloppillil[7]:

"...Linux and other open source software advocates are making a progressively more credible argument that OSS software is at least as robust – if not more – than commercial alternatives... the intrinsic parallelism and free idea exchange on OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mind-share threat... the ability to collect and harness the collective IQ of thousands of individuals across the Internet is simply amazing. More importantly, OSS evangelization scales with the size of the Internet much faster than our own evangelization efforts appear to scale..."

No se podría haber expresado mejor la lógica y la potencial superioridad del modelo *open source*. Naturalmente, Microsoft quería luchar contra este corriente pero sus tradicionales (y no siempre legales) prácticas monopolísticas se formulaban contra empresas, y no eran útiles para combatir la comunidad *open source*. Verdaderamente, los documentos de Halloween reconocían que Microsoft adoptaba una postura "a la defensiva" respecto del movimiento *open source* dado que, más que competir contra una empresa, se competía contra un proceso. De hecho, el modelo *open source* rompe con la lógica empresarial convencional. En este mercado el poder cambia de manos, abandonando los suministradores para apoyarse en los propios clientes. Pero todavía es posible combinar negocio (generación de ingresos) y *open source*, hay distintos caminos:

[7] La página web de Eric S. Raymond incluye un apartado relativo a esta documentación. Se puede consultar el texto en: http://www.catb.org/~esr/halloween/ [Fecha de consulta: 20 de junio de 2006]

- Modelos genéricos de negocio basados en la comercialización de paquetes útiles de programas *open source* y sus aplicaciones, con o sin programas propietarios. Esta opción puede incluir soporte técnico, servicios e instrucción. Se trata de el caso, entre otros, de Red Hat, VA Software, Bitkeeper o Hewlett Packard.
- Liberación del código fuente como una estrategia para popularizar un programa. Este fue el caso de Netscape.
- Construcción de un sistema propietario partiendo de soluciones *open source*. Mac OS X (Apple) es un ejemplo. La estrategia subyacente consiste en el hecho que los desarrolladores cambian de Windows a MAC OS porque tienen acceso a la fuente.
- Usar el potencial de la comunidad mundial de desarrolladores como un medio para aumentar el nivel de los productos de una compañía. Esta es la estrategia que ha seguido IBM, que ofrece todas sus mayores aplicaciones a la base de la plataforma Linux. Además, la compañía también ofrece servicios web basados en *open source* para competir con Microsoft.net
- Liberar el código fuente con algunas restricciones para beneficiarse del apoyo de la comunidad pero regulando algunas condiciones que beneficien a la propia compañía. Este ha sido el caso de Sun Microsystems con Java y posteriormente con Solaris. La comunidad *open source* se ha mostrado ambivalente respecto de estrategias de este tipo.

Quizá el modelo de desarrollo de *open source* más cuestionable es aquél que pretende un cambio de la organización del trabajo y de las relaciones de producción. Examinaremos este punto más adelante una vez visto con más detalle cómo funciona el modelo *open source*.

¿Cómo se trabaja en *open source* ?

Open source es un proceso de producción de conocimiento llevado a cabo por una comunidad que ha aprovechado el poder comunicativo y colaborativo de trabajar en red, fundamentalmente mediante Internet. *Open source* representa cuatro grandes cambios en

contraste con la fórmula tradicional de organización de la producción propia de una economía capitalista[8]:

1. La motivación individual. ¿Por qué programadores altamente cualificados contribuyen a los proyectos (tiempo y esfuerzo) sin recibir ninguna compensación?
2. ¿Cuál es la lógica económica que parte de la lógica convencional de mercado?
3. Coordinación. ¿Cómo se organizan cientos de individuos para cooperar libremente en un proyecto que no tiene una estructura hierárquica que organice la división del trabajo? ¿Cómo se implementa esta coordinación en los mecanismos externos de un mercado basado en la toma de decisiones jerárquica?
4. Administración de la complejidad. El desarrollo de software es una actividad altamente compleja que no se soluciona con un refuerzo de la mano de obra. De hecho, el estudio clásico llevado a cabo por Frederick Brooks[9] muestra que aumentando el número de programadores crecen las dificultades para completar satisfactoriamente un programa (Ley de Brooks). Esto ocurre porque con un número creciente de programadores el volumen de trabajo llevado a cabo crece linealmente mientras que la complejidad del proceso y la vulnerabilidad hacia posibles errores lo hace en progresión geométrica. Bajo estas condiciones la cuestión es: ¿cual es el procedimiento de gobierno que permite a la comunidad de programadores adquirir la calidad esperada de un programa en un proceso de trabajo con una complejidad semejante?

Veamos a continuación estos puntos de manera individualizada.

Motivación individual

Existen distintas encuestas, recogidas en el libro de Steve Weber *The success of open source*, sobre las motivaciones de los programadores vinculados a Linux. Aunque ciertamente

[8] WEBER, Steven. *The success of open source*. (p.133-134)
[9] BROOKS, Frederick. *The mythical man month: essays on software engineering*. (1974)

no son datos representativos del conjunto de la comunidad, la imagen de un desarrollador típico de Linux que se desprende de estos estudios es la de un individuo que se siente parte de una comunidad tecnológica, que quiere mejorar sus habilidades en programación, beneficiarse de un software superior y divertirse. No se preocupa por los ingresos económicos y, de hecho, se preocupa más por el tiempo que invierte en las tareas. Las mayores motivaciones son el aprendizaje individual, la eficiencia en el trabajo y la diversión.

Una encuesta llevada a cabo en 2001 por Boston Consulting Group[10] permitió categorizar a los desarrolladores de Linux en cuatro grandes grupos:

a) Creyentes. Motivados principalmente por la convicción que el software debería ser de código fuente abierto (1/3 de las respuestas)

b) Profesionales. Usan *open source* porque es una herramienta útil para sus respectivos trabajos (1/5 de las respuestas)

c) *Fun-seekers* (*busca-diversión*). Ven en el fenómeno *open source* una fuente de estimulación intelectual (1/4 de las respuestas)

d) *Skill-enhancers* (*habilidosos*). Aquellos a quienes el *open source* les permite progresar y convertirse en mejores programadores (1/5 de las respuestas).

La mayoría de los observadores, asumiendo los razonamientos de Eric S. Raymond, insisten en la importancia de la reputación entre la comunidad de programadores como estímulo y motivación. Esto no difiere demasiado de otras comunidades como por ejemplo la universitaria. El reconocimiento como igual por parte de aquellos que uno respeta es una de las mayores recompensas. Otro factor importante de motivación es la creencia que la comunidad potencia en los individuos la capacidad de autoayuda. Es una especie de individualismo comunitario. Así mismo, también fomenta la creencia en la innovación y la experimentación como uno de los potenciales más valorados de la conducta humana.

La lógica económica de un bien comunitario

[10] WEBER, Steven. (p. 135)

La clave en el proceso de cooperación es la libertad para entrar y salir de los proyectos. Así, el derecho al *forking* está abiertamente reconocido en la comunidad *open source*. *Forking* implica que en un momento dado cualquiera puede decidir que no comparte las decisiones tomadas acerca del código y empezar una línea nueva de programación que, en muchas ocasiones, termina siendo incompatible con el software que se está desarrollando en la comunidad. El *forking* está perjudicando a la comunidad porque dispersa esfuerzos y recursos, pero es esencial para los miembros saber que siempre pueden tomar esta alternativa. No obstante, ¿Cómo se consigue la coordinación?

En primer lugar, señalar las normas culturales en la comunidad. Esto se concreta en el derecho de propiedad del código fuente, implicando el derecho a distribuir versiones modificadas de software. La propiedad de este tipo se puede adquirir de tres formas: empezando a construir el programa (el fundador), recibiendo explícitamente la propiedad por parte del fundador, o retomando un proyecto que ha sido abandonado.

En segundo lugar, los mecanismos de toma de decisión implican un cierto tipo de hierarquía globalmente aceptada respecto del fundador, los respectivos programadores de ciertas líneas de código básicas para el desarrollo del proyecto, un núcleo de mantenedores del programa, y un grupo más amplio de desarrolladores acreditados (públicamente recogidos en las listas de correo del proyecto.) Esta hierarquía se desarrolla espontáneamente y está esencialmente basada en el reconocimiento de la capacidad técnica de los cooperadores. Así, en el corazón del *open source* podemos encontrar cierta racionalidad técnica. Sin embargo, la habilidad para mantener a la comunidad en crecimiento y dispuesta a colaborar está vinculada también a la capacidad de liderazgo.

El líder de un proyecto debe ser carismático y respetado, pero al mismo tiempo debe respetar a cada uno de los contribuidores y aceptar las críticas, tanto técnicas como personales. Los cambios en la comunidad pueden ser muy duros, pero al final se tiende a respetar la libertad de cada individuo **(***forking*). Lo más importante en el ejercicio del liderazgo es que todas las comunicaciones son públicas, de este modo la comunidad siempre

puede aportar su opinión. La contribución más importante de Linus Torvalds no fue tanto su kernel como su habilidad para construir una comunidad de cooperación basada en Internet. La mayor crisis para Linux sucedió en 1996 cuando Torvalds se mudó a Silicon Valley para trabajar en Transmeta, al mismo tiempo que se hacía cargo de sus hijas. Asumió un volumen de responsabilidades tan grande que descuidó las peticiones de la comunidad al no poder atender el conjunto de consultas que recibía diariamente. Las críticas fueron que Torvalds (no Linux) no sabía escalar… Su respuesta fue, después de algunos episodios agrios, descentralizar la toma de decisiones vinculadas a algunas partes del código, aunque mantuvo la última palabra respecto ciertos aspectos técnicos para poder preservar la unidad de Linux así como el conocimiento acumulado.

El liderazgo se entiende pues, como una combinación de "dictadura benevolente", descentralización en la toma de decisiones y confianza e interés en la cooperación de la comunidad, entendida como el único modo de escribir buen software. Ciertamente existen *egos* implicados en el proceso, y esto es algo normal y globalmente aceptado. Lo que no se tolera es el uso de una posición de poder en beneficio propio (apropiarse del trabajo de otros sin liberar resultados…) Si hay gente que gana dinero por su cuenta, no hay problema siempre que no bloquee información o mejoras que son producidas colectivamente en la comunidad. No se trata de una comunidad de "ángeles" o de activistas contraculturales, se trata de una cooperativa de programadores y de técnicos que son conscientes de que el software *open source* funciona igual o mejor que el propietario y, además, disfrutan de la oportunidad de innovar y de ser respetados por sus compañeros.

Administración de la complejidad

El software es un producto extremadamente complejo. La versión 7.1 que Red Hat hizo de Linux tiene más de 30 millones de líneas de código. En organizaciones formales el proceso de producción de una tarea de tales dimensiones implicaría una división compleja del trabajo, con un amplio número de programadores que coordinar, induciendo gran cantidad de problemas vinculados a la eficiencia (conviene recordar el análisis hecho por Brook). Este es, de hecho, el caso de Microsoft, y esa es la razón por la cual el software que producen está

repleto de fallos, lo que los programadores libres llaman "blue screen of doom" (pantalla azul del destino).

La clave para reducir los problemas vinculados a la complejidad es la modularización, en el sentido de diferenciación de tareas por subconjuntos de problemas (manteniendo la compatibilidad del software). Pero resulta difícil planificar con antelación quien hace qué en la medida que se trata de un proceso de innovación. Precisamente es mediante el ensayo y error que un programa avanza. La cooperación del *open source* en Internet hace transparente el progreso de un programa y las líneas de desarrollo adoptadas por los subconjuntos que se concentran en módulos específicos. Así, cada uno de los grupos de la comunidad tiene la responsabilidad de solucionar problemas concretos más que tratarse de la ejecución de tareas preplanificadas.

Administración de la cooperación

Una cooperación disciplinada se consigue mediante la sanción o el escarnio público de aquellos miembros de la comunidad que no se ciñen a las normas y reglas implícitas o explícitas. No obstante, es necesario algo más: una estructura formal para proyectos a gran escala. Esto varía en función de cada proyecto. Por ejemplo, Apache, que empezó con 8 personas en 1995, creció rápidamente hasta llegar a un núcleo de algunas docenas de desarrolladores distribuidos alrededor del mundo. Apache organizó un sistema político en el que cualquiera puede expresar su opinión, pero las únicas que son vinculantes son aquellas que se aprueban por el conjunto del grupo. Los miembros son elegidos en base a sus méritos técnicos y sus contribuciones al programa (evaluadas por *peer review*, algo que no dista demasiado de la Academia). Existe el poder de veto en aquellas decisiones que afectan directamente el programa: cualquier cambio en el código requiere al menos 3 votos a favor y ningún voto en contra. Pero cualquier voto negativo debe estar argumentado. En 1999 el Grupo Apache adoptó la estructura legal de *Apache Software Foundation*, que proporcionaba a su vez un paraguas institucional para otros proyectos vinculados al panorama web como por ejemplo Jakarta, Perl, TCL …

Linux tiene una estructura hierárquica semi-formal. En cabeza está, naturalmente, Linus Torvalds pero a su alrededor originalmente había los "ancianos de la tribu" (en aquel momento mayoritariamente eran menores de 30 años) nombrados en base a sus contribuciones al código. En 1994 Torvalds hizo pública una lista de 80 desarrolladores en un archivo oficial con los créditos de Linux. En 1996 añadió una lista de aquellos que prestaban mantenimiento al proyecto (un desarrollador encargado del mantenimiento es aquél programador que se responsabiliza de un segmento específico del código). Posteriormente, Torvalds designó algunos "segundos" con un grado de responsabilidad mayor. El reconocimiento formal de estatus en estas listas contribuye a la reputación de los programadores y, eventualmente, ayuda en el panorama laboral externo a la comunidad de Linux. A cambio, éstos respetan la organización del trabajo. Algunas veces una excesiva centralización de la toma de decisiones en Linux se convierte en algo poco funcional que puede llevar a la rebelión, como el episodio vivido en 2002. Sin embargo, remplazar a Torvalds como la última autoridad podría desorganizar la red de trabajo cooperativo y parar el desarrollo de Linux. Entre los años 1991 y 2005 Linux ha evolucionado con cientos de desarrolladores fijos colaborando en el programa, y miles de colaboradores ocasionales, sin haberse producido demasiadas interrupciones y sin una organización excesivamente formal.

Open source, organización cooperativa de la producción

Open source, como hemos mencionado anteriormente, es un modo de organizar la producción, desafiando formas tradicionales de la división del trabajo, hierarquías organizacionales, y derechos de propiedad convencionales. Distintos análisis exponen la posibilidad de extender este tipo de organización a otras áreas más allá del software, basándose en los principios que caracterizan el proceso del *open source*:

- Innovación *user-driven* en canales paralelos de distribución
- Conducta cooperativa regulada por normas culturales y reglas de gobierno
- Lógica económica basada en la no-exclusión, la no-rivalidad y sinergias de trabajo en red

- Redefinición de la noción de derechos de propiedad. Los derechos de propiedad en *open source* se construyen en base al derecho a distribuir, y a no excluir. Conviene recordar que la propiedad es un concepto construido socialmente. La experiencia con los derechos de propiedad intelectual en ámbitos como la distribución musical en Internet es una ilustración a tener en consideración de este principio, y también de las contradicciones implícitas.
- Internet permite y aumenta este nuevo sistema de cooperación, mientras crea serias dificultades para el refuerzo de los derechos de propiedad tradicionales.

La expansión del *open source* hacia otros ámbitos de actuación se basa en la implementación de cuatro principios:

a) Autorizar a la gente a que experimente y se dote con la tecnología apropiada (con los pertinentes incentivos sociales).

b) Encontrar una solución de ingeniería para que los bits de información se encuentren a sí mismos.

c) Estructurar la información para que se pueda recombinar con otras piezas de información (modularización).

d) Crear un sistema de gobierno que sustente el proceso (la lógica de la GPL es un ejemplo de institucionalización de nuevos derechos de propiedad).

Open source, tecnología, y desarrollo mundial

El *open source* implica un desafío de los usos oligopolísticos de los derechos de propiedad intelectual (incluida la tecnología), alienta la difusión de conocimiento y la innovación en las aplicaciones del conocimiento existente, adaptado a las necesidades de los usuarios. Esta es la razón principal por la cual el *open source* ha generado esperanza en muchos países con altos índices de programadores puesto que los desarrolladores pueden potenciar su capacidad creando programas orientados a sus necesidades sin sentirse atados por los derechos de propiedad de las corporaciones multinacionales. Además, incorporando en las

redes de cooperación un gran número de programadores provenientes del mundo del desarrollo de software, y afilando sus capacidades en el marco de la comunidad, el mundo en su globalidad estará en condiciones de experimentar un mayor poder de innovación. Este es el principal motivo por el que un creciente número de países en desarrollo (Brasil, India, aunque también China…) están adoptando el *open source* y experimentando su potencial. Un ejemplo es el proyecto Simputer (India), con portátiles que corren sobre Linux (cuestan 250 US $) diseñados por el Instituto Indio de Ciencia (Indian Institute of Science) y la compañía bengalí Encore. China también ha desarrollado *Red Flag Linux* y otros paquetes de desarrollo. Microsoft está lanzando una gran campaña en este país y otros para contrarrestar la expansión del *open source*, pero la difusión del *open source* está arraigada en las necesidades de desarrollo de países y organizaciones a nivel mundial. Las aplicaciones del *open source* en el software y la computación podrían tener efectos realmente significativos para la salud y la educación en los países en desarrollo, por ejemplo en la asistencia primaria. Así, la batalla ideológica y comercial sobre el uso del *open source* es un tema candente cuando hoy en día se habla sobre desarrollo.

Como hemos visto anteriormente el software libre y el software de código abierto son conceptos que habitualmente se confunden y se usan indistintamente. Ambas definiciones son con frecuencia confusas particularmente en lo referente al coste y a la comercialización de los productos, aunque compartan una misma base. Las dos iniciativas principales son la *Free Software Foundation* (FSF), fundada el 1984 por Richard Stallman, y la *Open Source Initiative* (OSI), fundada el 1998 por Bruce Perens y Eric S. Raymond[11].

Los principales puntos de discordia entre dichos movimientos son los concernientes a la distribución y a la llamada "cláusula viral". Mientras que la OSI mantiene un enfoque comercial de los productos y concibe la liberación del código como un aspecto más, la FSF mantiene la cláusula viral y la gratuidad[12] como elementos distintivos del movimiento elevándolos al estatus de "filosóficos". Los principales ideales de la FSF se concentran en lo que Stallman denominó "libertades básicas" y que anteriormente hemos recogido[13].

Los criterios recogidos por la OSI para que un software sea considerado *open source* se basan en un documento titulado "The Debian Free Software Guidelines" elaborado por el propio Perens el año 1997. Los puntos recogidos son los siguientes[14]:

1- Redistribución libre de los productos (con o sin coste adicional)
2- Acceso al código fuente.
3- Trabajos derivados del original o modificaciones de éste pueden ser redistribuidos bajo los mismos términos.
4- Integridad del código fuente del autor: algunas licencias pueden restringir la difusión del código a parches (*patch files*).
5- No discriminación de personas o colectivos: la licencia no puede excluir a nadie.

[11] Eric S.Raymond es el ideólogo de la teoría de la "Catedral y el bazar", comprendida en un ensayo publicado bajo el mismo título el año 1997.
[12] Se contempla la posibilidad de cobrar tasas módicas en concepto de distribución.
[13] www.fsf.org
[14] http://www.opensource.org/docs/definition.html [Fecha de consulta: 20 de junio de 2006]

6- No discriminación de áreas de iniciativa: se respeta la posibilidad de una distribución comercial del software.
7- Distribución de la licencia: los derechos vinculados al programa aplican a todo aquel que lo reciba.
8- La licencia no debe ser específica de un producto: aunque una parte del programa se distribuyera separadamente, los derechos asignados son los mismos que los del conjunto original.
9- La licencia no debe restringir otro software: el acceso al código no es vinculante a otros productos de software que se deriven del original.
10- La licencia debe ser tecnológicamente neutral: no debe requerirse la aceptación de la licencia por medio de un acceso o interfície específica.

La FSF y la OSI son dos de las iniciativas que lideran los movimientos de código abierto, pero hay otras instituciones que prestan su apoyo, tanto en el apartado relativo a infraestructura como comunicación y búsqueda de recursos. Algunas empresas también colaboran e incentivan el desarrollo de software de código abierto. *Sun Microsystems*, por ejemplo, es una de las corporaciones que más ha contribuido liberando el 13 de octubre del año 2000 el paquete ofimático *Star Office*, que en versión libre pasó a denominarse *Open Office*. Aunque el uso de software libre y de fuentes abiertas en el entorno escritorio es muy limitado, como veremos más adelante, dicha contribución junto al hecho que se desarrollara un entorno gráfico atractivo de escritorio (Gnome / KDE) ha sido clave para la popularización del sistema GNU/Linux.

Las creaciones vinculadas al sector del software de código fuente abierto se pueden regular mediante licencias. Éstas pueden variar según los matices con los que los programadores quieran dotar a su software. Así, coexisten multiplicidad de licencias de entre las cuales la más popular y la que acoge a más proyectos es la GNU/GPL (GNU General Public License), desarrollada por Stallman y vinculada a la FSF. La siguiente tabla recoge las

cinco licencias más utilizadas y el número de proyectos adscritos[15]. Se puede observar que las diferencias son más que notables:

Licencia	Nombre	Proyectos
GPL	GNU General Public License	6117
LGPL	GNU Lesser General Public License	835
BSD	Berkeley Software Distribution	478
Artistic	The "Artistic License"	298
MPL	Mozilla Public License	143

Otros términos vinculaos al software de código fuente abierto son el "shared source", el "freeware" y el "shareware". Los tres comparten elementos concernientes al código fuente pero lejos de matizar conceptos, agrandan la confusión entre el público no familiarizado con el software de código fuente abierto. La diferencia básica entre "freeware" y "shareware" radica en el hecho que, aunque ambos se distribuyen sin coste económico este último incluye limitaciones de uso, y habitualmente tan sólo se puede ejecutar durante un periodo corto de tiempo. En el caso de "freeware", no es frecuente que se distribuya con el código fuente, pero su uso es gratuito e ilimitado.

Aunque el programa Shared Source no es una exclusiva de Microsoft, es la compañía que más lo ha popularizado. Se implanta en la corporación el año 2001 como respuesta a la creciente presión que hacen los movimientos defensores del código abierto, bajo el acrónimo MSSI (*Microsoft's Shared Source Initiative*)[16]. El modelo de negocio de Microsoft, como empresa de software comercial, parte de la restricción al acceso al código fuente de su sistema operativo así como de sus programas, aunque bajo ciertas licencias admite la posibilidad que se pueda acceder a dichos datos (sólo de algunos productos[17]). Así, existen varios programas englobados por la iniciativa *Shared Source* aunque el acceso es restrictivo y se ciñe a la

[15] 2001, Programa IDA.
[16] http://www.microsoft.com/resources/sharedsource/default.mspx [Fecha de consulta: 20 de junio de 2006]
[17] Windows 2000, Windows XP, Windows server 2003, Windows CE, Windows CE Premium, ASP.NET Samples, C#Jscript/CLI Implementations, ConferenceXP, FlexWiki, IronPyton, Smart Devices Developer Samples, Source Tools for Bluetooth, Visual Studio.NET Academic Tools, WiX y WTL.

consulta de las líneas de código. Éste no sólo no se puede modificar sino que tampoco no se permite ni la comercialización ni la libre distribución de productos derivados. El acceso al código fuente pues, no sigue los parámetros recogidos anteriormente por la FSF o la OSI sino que se adapta al modelo de negocio propio de la compañía.

Otra iniciativa llevada a cabo por Microsoft y que permite el acceso al código fuente de sus productos (también se refiere únicamente a su consulta) es el Programa de Seguridad para Gobiernos (GSP – *Government Security Program*). Se contabilizan 25 países adscritos, uno de los cuales es España, aunque Microsoft asegura que hay otros que no lo han hecho público pese estar adheridos. El convenio con el gobierno español se firmó en enero de 2004, su vigencia es de tres años y su aplicación concierne exclusivamente al Centro Nacional de Inteligencia (CNI). Éste tendrá acceso al código fuente de *Windows* así como a toda la información técnica que requiera para poder revisar las características del sistema en lo que a seguridad se refiere.

Las diferencias implícitas en ambos modelos de negocio (libre/propietario) derivan con frecuencia en enfrentamientos dialécticos. Uno de los episodios más tensos entre la comunidad *open source* y Microsoft, a parte de las numerosos altercados ocasionadas por las comisiones antimonopolio y la recientemente rechazada ley europea de patentabilidad del software, fue el que se vivió en mayo de 2001 con motivo de una conferencia pronunciada por uno de los vicepresidentes de Microsoft Corporation, Craig Mundie, en la New York University Stern School of Business[18]. La charla, que trató de los modelos de gestión económica de los negocios en la era de Internet y de las comunicaciones, incluyó en varios momentos menciones explícitas a las comunidades *open source* y software libre. Mundie calificó al modelo del software de código fuente abierto de "inviable e insostenible", comparándolo al fracasado modelo de las empresas *punto com*. La respuesta no tardó en llegar

[18] http://www.microsoft.com/presspass/exec/craig/05-03sharedsource.mspx [Fecha de consulta: 20 de junio de 2006]

y los principales líderes de la "comunidad"[19] presentaron un documento conjunto reivindicando la unidad de la comunidad libre bajo el título "Free software leaders stand together"[20]. Junto al mensaje de unidad implícito en la propia redacción del texto, el artículo finaliza con una llamada a Microsoft para que se una a la comunidad, abriendo el código y abandonando iniciativas restrictivas como la MSSI. La elaboración y firma de dicho documento también llevaba implícito un mensaje de unión pese a la disparidad propia de una comunidad tan grande y diversa.

Paralelamente, el pasado mes de abril se celebró en Luxemburgo el juicio que mantiene enfrentados a Microsoft y a la Unión Europea por una supuesta violación de las leyes de libre competencia desde 2004. Se acusa a la multinacional de cometer un abuso de posición dominante en dos apartados distintos: la venta del lector multimedia *Windows Media Player* junto al sistema operativo Windows, y la negativa de proporcionar a otras empresas del sector información técnica para garantizar la interoperabilidad en el mercado de servidores de grupos de trabajo (protocolos de comunicación cliente/servidor y servidor/servidor). Actualmente Microsoft comercializa una versión de su sistema operativo que no incluye el reproductor citado anteriormente (una de las medidas correctoras) pero la Comisión Europea sigue acusando a la multinacional de no haber facilitado la información relativa a interoperabilidad, algo que no debe confundirse con el código fuente de Windows. Microsoft se enfrenta a una multa diaria de dos millones de euros hasta que cumpla el otro de los requisitos a los que le obliga la sentencia de marzo de 2004. La multinacional alega que la difusión de esta información técnica supondría un daño irreparable para su estrategia empresarial pero la documentación que ha presentado al respecto se considera insuficiente. Al cierre de este informe todavía no se conocía la resolución final del conflicto aunque distintos medios de comunicación señalan que la salida de la nueva versión de Windows (*Windows Vista*) podría conllevar problemas semejantes.

[19] Richard Stallman, Bruce Perens, Eric Raymond, Linus Torvalds, Miguel de Icaza, Larry Wall, Guido van RFLOSSum, Tim O'Reilly, Bob Young y Larry Augustin.
[20] http://perens.com/Articles/StandTogether.html [Fecha de consulta: 20 de junio de 2006]