1. Introducción

> "If you have an apple and I have an apple and we exchange apples, then you and I will still each have one apple. But if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas."
>
> "Si tú tienes una manzana y yo tengo una manzana y las intercambiamos, seguiremos teniendo una manzana cada uno. Pero si tú tienes una idea y yo tengo una idea y las intercambiamos, cada uno de nosotros tendrá dos ideas."
>
> Atribuido a Bernard Shaw

¿Qué es el software libre? ¿Qué es y qué implicaciones tiene la licencia de un programa libre? ¿Cómo se está desarrollando el software libre? ¿Cómo se financian los proyectos de software libre y qué modelos de negocio relacionados con ellos se están experimentando? ¿Qué motiva a los desarrolladores, especialmente a los que son voluntarios, a involucrarse en proyectos de software libre? ¿Cómo son estos desarrolladores? ¿Cómo se coordinan en sus proyectos y cómo es el software que producen? En resumen, ¿cuál es el panorama general del software libre? Éste es el tipo de preguntas que trataremos de responder en este texto. Porque aunque el software libre está cada vez más presente en los medios de comunicación y en las conversaciones de los profesionales de la informática, y aunque incluso empieza a estar en boca de los ciudadanos en general, aún es un desconocido para muchos. Y los que lo conocen muchas veces no saben más que de algunos de sus aspectos, y desconocen completamente otros.

Para empezar, en este capítulo vamos a presentar los aspectos específicos del software libre, centrándonos fundamentalmente en explicar sus bases para los que se aproximen al tema por primera vez y en motivar su importancia. Entre estas bases nos detendremos en la definición del término (para saber de qué vamos a hablar) y en las consecuencias principales del uso (y de la mera existencia) del software libre.

1.1. El concepto de *libertad* en el software

Desde principios de los años setenta nos hemos acostumbrado a que quien comercializa un programa pueda imponer (e imponga) las condiciones bajo las que puede usarse. Puede, por ejemplo, prohibir que sea prestado a un tercero. A pesar de que el software es el elemento tecnológico más flexible y adaptable que tenemos, puede imponerse (y es común imponer) la imposibilidad de adaptarlo a unas necesidades concretas, o de corregir sus errores, sin el permiso explícito del productor, que normalmente se reserva en exclusiva estas posibilidades. Pero ésta es sólo una de las posibilidades que ofrece la legislación actual: el *software libre*, por el contrario, otorga las libertades que el *software privativo* niega.

Software privativo

En este texto utilizaremos el término *software privativo* para referirnos a cualquier programa que no pueda considerarse software libre, de acuerdo con la definición que se ofrece más adelante.

1.1.1. Definición

Así pues, el término *software libre* (o *programas libres*), tal como fue concebido por Richard Stallman en su definición (Free Software Foundation, "Free software definition" –http://www.gnu.org/philosophy/free-sw.html– [120]), hace referencia a las libertades que puede ejercer quien lo recibe, concretamente cuatro:

1) Libertad para ejecutar el programa en cualquier sitio, con cualquier propósito y para siempre.

2) Libertad para estudiarlo y adaptarlo a nuestras necesidades. Esto exige el acceso al código fuente.

3) Libertad de redistribución, de modo que se nos permita colaborar con vecinos y amigos.

4) Libertad para mejorar el programa y publicar sus mejoras. Esto también exige el código fuente.

El mecanismo que se utiliza para garantizar estas libertades, de acuerdo con la legalidad vigente, es la distribución mediante una licencia determinada, como veremos más adelante (capítulo 3). En ella el autor plasma su permiso para que el receptor del programa pueda ejercer esas libertades, y también las restricciones que pueda querer aplicar (como dar crédito a los autores originales en caso de redistribución). Para que la licencia sea considerada libre, estas restricciones no pueden ir en contra de las libertades mencionadas.

La ambigüedad de *free*

El término original en inglés para *programas libres* es *free software*. Sin embargo, *free*, además de 'libre', significa 'gratis', lo que genera gran confusión. Por ello a menudo en inglés se toman prestadas palabras españolas y se habla de *libre software*, en contraposición a *gratis software*, al igual que nosotros tomamos prestada la palabra *software*.

Así pues, las definiciones de software libre no hacen ninguna referencia a que pueda conseguirse gratuitamente: el software libre y el software gratuito son cosas bien distintas. Sin embargo, dicho esto, hay que explicar también que debido a la tercera libertad, cualquiera puede redistribuir un programa sin pedir contraprestación económica ni permiso, lo que hace prácticamente imposible obtener grandes ganancias simplemente por la distribución de software libre: cualquiera que lo haya obtenido puede a su vez redistribuirlo a precio más bajo, o incluso gratis.

Nota

A pesar de que cualquiera puede comercializar un programa dado a cualquier precio, y eso hace que teóricamente el precio de redistribución tienda hacia el coste marginal de copia, existen modelos de negocio basados precisamente en vender software, porque hay muchas circunstancias en las que el consumidor está dispuesto a pagar si recibe ciertas contraprestaciones, como por ejemplo una cierta garantía, aunque sea subjetiva, sobre el software que adquiere o un valor añadido en forma de selección, actualización y organización de un conjunto de programas.

Desde un punto de vista práctico, hay varios textos que definen más precisamente qué condiciones tiene que cumplir una licencia para ser considerada como de software libre. Entre ellos, destacan por su importancia histórica la definición de software libre de la Free Software Foundation (http://www.gnu.org/philosophy/free-sw.html) [120], las directrices de Debian para decidir si un programa es libre (http://www.debian.org/social_contract.html#guidelines) [104] y la definición del término *open source* por la Open Source Initiative (http://www.opensource.org/docs/definition_plain.html) [215], muy similar a las anteriores.

Nota

Por ejemplo, las directrices de Debian entran en el detalle de permitir que el autor exija que los códigos fuente distribuidos no sean modificados directamente, sino que los originales se acompañen de parches separados, y que los programas binarios se generen con nombres distintos del original. Además exigen que las licencias no contaminen otros programas distribuidos en el mismo medio.

1.1.2. Términos relacionados

Equivalente de *software libre* es el término *open source software* ('programas de fuente abierta'), promovido por Eric Raymond y la Open Source Initiative. Filosóficamente, el término es muy distinto, ya que hace énfasis en la disponibilidad de código fuente, no en la libertad, pero su definición es prácticamente la misma que la de Debian ("The open source definition", 1998 –http://www.opensource.org/docs/definition_plain.html–) [183]. Este nombre es más políticamente aséptico y recalca un aspecto técnico que puede dar lugar a ventajas técnicas, como mejores modelos de desarrollo y negocio, mayor seguridad, etc. Fuertemente criticado por Richard Stallman ("Why *free software* is better than *open source*") [204] y la Free Software Foundation (http://www.fsf.org) [27], ha encontrado mucho más eco en la literatura comercial y en las estrategias de las empresas que de una manera u otra apoyan el modelo.

Otros términos relacionados de algún modo con el software libre son los siguientes:

Freeware	Son programas gratuitos. Normalmente se distribuyen sólo en binario, y se pueden obtener sin coste. A veces se consigue también permiso de redistribución, pero otras no, de manera que entonces sólo se pueden obtener del sitio "oficial" mantenido a ese efecto. Es habitual que se usen para promocionar otros programas (típicamente con funcionalidad más completa) o servicios. Ejemplos de este tipo de programas son Skype, Google Earth o Microsoft Messenger.
Shareware	No es siquiera software gratis, sino un método de distribución, ya que los programas, generalmente sin códigos fuente, se pueden copiar libremente, pero no usar continuadamente sin pagarlos. La exigencia de pago puede estar incentivada por funcionalidad limitada, mensajes molestos o una simple apelación a la moral del usuario. Además, las estipulaciones legales de la licencia podrían utilizarse en contra del infractor.
Charityware, careware	Se trata generalmente de *shareware* cuyo pago se pide para una organización *caritativa* patrocinada. En muchos casos, el pago no se exige, pero se solicita una contribución voluntaria. Algún software libre, como **Vim**, solicita contribuciones voluntarias de este tipo (Brian Molenaar, "What is the context of charityware?") [173].
Dominio público	El autor renuncia absolutamente a todos sus derechos en favor del común, lo cual tiene que quedar explícitamente declarado en el programa, ya que si no se dice nada, el programa es propietario y no se puede hacer nada con él. En este caso, y si además se proporcionan los códigos fuente, el programa es libre.
Copyleft	Se trata de un caso particular de software libre cuya licencia obliga a que las modificaciones que se distribuyan sean también libres.
Propietario, cerrado, no libre	Se trata de términos usados para denominar al software que no es libre ni de fuente abierta.

1.2. Motivaciones

Como hemos visto, hay dos grandes familias de motivaciones para el desarrollo de software libre, que dan lugar asimismo a los dos nombres con que se lo conoce:

- La motivación ética, abanderada por la Free Software Foundation (http://www.fsf.org) [27], heredera de la cultura *hacker* y partidaria del apelativo *libre*, que argumenta que el software es conocimiento que debe poder difundirse sin trabas y cuya ocultación es una actitud antisocial, y que afirma además que la posibilidad de modificar programas es una forma de libertad de expresión. Puede profundizarse en este aspecto en los ensayos de Stallman (*Free software, free society. Selected essays of Richard M. Stallman*) [211] o en el análisis de Pekka Himanen (*The hacker ethic and the spirit of the information age*. Random House, 2001) [144].

- La motivación pragmática, abanderada por la Open Source Initiative (http://www.opensource.org) [54] y partidaria del apelativo *fuente abier-*

ta, que argumenta ventajas técnicas y económicas que repasaremos en el apartado siguiente.

Aparte de estas dos grandes motivaciones, la gente que trabaja en software libre puede hacerlo por muchas otras razones, que van desde la diversión (Linus Torvalds y David Diamond, Texere, 2001) [217] hasta la mera retribución económica, posiblemente debida a *modelos de negocio* sustentables. En el capítulo 4 se profundiza en estas motivaciones a partir de análisis objetivos.

1.3. Consecuencias de la libertad del software

El software libre trae consigo numerosas ventajas y pocas desventajas, muchas de las cuales han sido exageradas (o falseadas) por la competencia propietaria. De ellas, la que más fundamento tiene es la económica, ya que como hemos visto, no es posible obtener mucho dinero de la distribución y ésta la puede y la suele hacer alguien distinto del autor. Por ello se necesitan modelos de negocio y otros mecanismos de financiación, que se desarrollan en el capítulo 5. Otras desventajas, como la falta de soporte o la calidad escasa, están relacionadas con la financiación, pero además en muchos casos son falsas, ya que incluso software sin ningún tipo de financiación suele ofrecer muy buen soporte gracias a foros de usuarios y desarrolladores, y muchas veces tiene gran calidad.

Teniendo presentes las consideraciones económicas, hemos de observar que el modelo de costes del software libre es muy distinto del modelo de costes del software propietario, ya que gran parte de él se ha desarrollado fuera de la economía formal monetaria, muchas veces con mecanismos de trueque: "yo te doy un programa que te interesa y tú lo adaptas a tu arquitectura y le haces mejoras que a ti te interesan." En el capítulo 7 se explican mecanismos de ingeniería de software apropiados para aprovechar estos recursos humanos no pagados y con características propias, mientras que en el capítulo 8 se estudian las herramientas usadas para hacer efectiva esta colaboración. Pero además, gran parte de los costes disminuyen por el hecho de que es libre, ya que los programas nuevos no tienen por qué empezar desde cero, sino que pueden reutilizar software ya hecho. La distribución tiene también un coste mucho menor, ya que se hace por Internet y con propaganda gratuita en foros públicos destinados a ello.

Otra consecuencia de las libertades es la calidad derivada de la colaboración voluntaria de gente que contribuye o que descubre y notifica errores en entornos y situaciones inimaginables por el desarrollador original. Además, si un programa no ofrece la calidad suficiente, la competencia puede tomarlo y mejorarlo partiendo de lo que hay. Así la *colaboración* y la *competencia*, dos poderosos mecanismos, se combinan para conseguir una mejor calidad.

Examinemos ahora las consecuencias beneficiosas según el destinatario.

1.3.1. Para el usuario final

El usuario final, ya sea individual o empresa, puede encontrar verdadera competencia en un mercado con tendencia al monopolio. En particular, no depende necesariamente del soporte del fabricante del software, ya que puede haber múltiples empresas, quizá pequeñas, que dispongan del código fuente y de conocimientos y que puedan hacer negocio manteniendo determinados programas libres.

Ya no se depende tanto de la *fiabilidad* del fabricante para intentar deducir la calidad de un producto, sino que la guía nos la dará la aceptación de la comunidad y la disponibilidad de los códigos fuente. Además, nos podemos olvidar de *cajas negras*, en las que hay que confiar "porque sí", y de las estrategias de los fabricantes, que pueden decidir unilateralmente dejar de mantener un producto.

La evaluación de productos antes de su adopción es ahora mucho más sencilla, ya que basta con instalar los productos alternativos en nuestro entorno real y probar, mientras que para software propietario hay que fiarse de informes externos o negociar pruebas con los proveedores, lo cual no siempre es posible.

Dada la libertad de modificar el programa para uso propio, el usuario puede personalizarlo o adaptarlo a sus necesidades, corrigiendo errores si los tuviera. El proceso de corrección de errores descubiertos por los usuarios en software propietario suele ser extremadamente penoso, si no imposible, ya que si conseguimos que se reparen, a menudo ello se hará en la versión siguiente, que puede tardar años en salir y que a veces, además, habrá que comprar de nuevo. En software libre, sin embargo, podemos hacer nosotros dicha reparación, si estamos cualificados, o contratar el servicio fuera. También podemos, directamente o contratando servicios, integrar el programa con otro o auditar su calidad (por ejemplo la seguridad). El control pasa, en gran medida, del proveedor al usuario.

1.3.2. Para la Administración pública

La Administración pública es un gran usuario de características especiales, ya que tiene obligaciones especiales con el ciudadano, sea proporcionándole servicios accesibles, neutrales respecto a los fabricantes, o garantizando la integridad, la utilidad, la privacidad y la seguridad de sus datos a largo plazo. Todo ello la obliga a ser más respetuosa con los estándares que las empresas privadas y a mantener los datos en formatos abiertos y manipulados con software que no dependa de estrategia de empresas, generalmente extranjeras, certificado como seguro por auditoría interna. La adecuación a estándares es una característica notable del software libre no tan respetada por el software propietario, generalmente ávido de crear mercados cautivos.

Asimismo, la Administración tiene una cierta función de escaparate y guía de la industria que le hace tener un gran impacto que debería dirigirse a la creación de un tejido tecnológico generador de riqueza nacional. Dicha riqueza puede crearse mediante el fomento de empresas cuyo negocio sea, en parte, el desarrollo de nuevo software libre para la Administración o el mantenimiento, la adaptación o la auditoría del software existente. En el capítulo 6 nos extenderemos más en esta cuestión.

1.3.3. Para el desarrollador

Para el desarrollador y productor de software, la libertad cambia mucho las reglas del juego. Con ella le es más fácil competir siendo pequeño y adquirir tecnología punta. Puede aprovecharse del trabajo de los demás, compitiendo incluso con otro producto mediante la modificación de su propio código, si bien también el competidor copiado se aprovechará de nuestro código (si es *copyleft*). Si el proyecto se lleva bien, puede conseguirse la colaboración gratuita de mucha gente, y además, se tiene acceso a un sistema de distribución prácticamente gratuito y global. No obstante, queda pendiente el problema de cómo obtener recursos económicos si el software realizado no es fruto de un encargo pagado. En el capítulo 5 se tratará en detalle este tema.

1.3.4. Para el integrador

Para el integrador, el software libre es el paraíso. Significa que ya no hay más cajas negras que intentar encajar, a menudo con ingeniería inversa. Puede limar asperezas e integrar trozos de programas para conseguir el producto integrado necesario, al disponer de un acervo ingente de software libre de donde extraer las piezas.

1.3.5. Para el que proporciona mantenimiento y servicios

Disponer del código fuente lo cambia todo y nos sitúa casi en las mismas condiciones que el productor. Si no son las mismas es porque hace falta un conocimiento profundo del programa que sólo el desarrollador posee, por lo que es conveniente que el mantenedor participe en los proyectos que se dedica a mantener. El valor añadido de los servicios es mucho más apreciado, ya que el coste del programa es bajo. Éste es actualmente el negocio más claro con software libre y con el que es posible un mayor grado de competencia.

1.4. Resumen

Este primer capítulo ha servido como toma de contacto con el mundo del software libre. El concepto, definido por Richard Stallman, se basa en cuatro libertades (libertad de ejecución, libertad de estudio, libertad de redistribución y libertad de mejora), dos de las cuales suponen el acceso al código fuente. Esta accesibilidad y sus ventajas motivan otro punto de vista menos ético y más pragmático, defendido por la Open Source Initiative, que ha dado lugar

a otro término: *software de fuente abierta*. Se han comentado también otros términos relacionados por similitud o contraposición, y que permiten aclarar los conceptos. Finalmente, se ha hablado de las consecuencias de la libertad del software para los principales actores implicados.