Software libre

PID_00269820

César Pablo Córcoles Briongos
Ismael Peña-López

Tiempo mínimo de dedicación recomendado: 1 hora

Universitat Oberta de Catalunya

César Pablo Córcoles Briongos

Licenciado en Matemáticas por la Universitat Autònoma de Barcelona. Profesor de los estudios de Informática, Multimedia y Telecomunicaciones desde 2001. Coordina asignaturas del ámbito del diseño y desarrollo web del programa de Grado en Multimedia. Director del máster universitario de Desarrollo de Sitios y Aplicaciones Web. Su área de interés en investigación se centra en el uso de recursos multimedia (animación, visualización 3D) e interactivos para la docencia de las ciencias, con especial atención a las materias STEM.

Ismael Peña-López

Profesor en Estudios de Derecho y Ciencias Políticas (UOC) e investigador en Internet Interdisciplinary Institute y en eLearn Center, también de la UOC. Doctor en Sociedad de la Información y del Conocimiento, licenciado en Ciencias Económicas y Empresariales (Economía), máster en Ecoauditorías y planificación empresarial del medio ambiente y posgraduado en Gestión del conocimiento. Trabaja sobre el impacto de las tecnologías de la información y la comunicación en el desarrollo. En concreto, los intereses se centran en la medida de la evolución de las economías digitales y la adopción personal de lo que es digital (*e-readiness*, divisoria digital), y también el impacto de las TIC en el desarrollo y sus principales instituciones, especialmente en el ámbito de las TIC y la educación y las TIC y la democracia.

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Iván Serrano Balaguer (2020)

Primera edición: febrero 2020

Av. Tibidabo, 39-43, 08035 Barcelona
Realización editorial: FUOC

Índice

Introducción

Igual que ocurre con los mensajes cifrados, en los que solamente el emisor y determinados receptores pueden ver un contenido inteligible, los programas informáticos están escritos en un lenguaje de programación que, por norma general, no puede ser interpretado por la mayoría de los usuarios que acceden al código de un programa.

A diferencia del caso de la encriptación, aquí la razón es técnica: mientras que los informáticos necesitan programar mediante lenguajes de programación que tengan una estructura y una serie de órdenes reconocibles por los humanos, los ordenadores no suelen –con la excepción de los lenguajes de interpretación y algún otro caso– entender dichos lenguajes, por lo que se hace necesario traducirlos al llamado lenguaje máquina, mediante el proceso de **compilación** de un programa. Este proceso hace que, a partir de entonces, existan dos versiones del mismo programa:

- El programa compilado, que es utilizado por el ordenador y resulta totalmente ininteligible para los humanos.
- El **código fuente**, que es el programa original antes de ser compilado y que se programó utilizando lenguajes comprensibles para los expertos.

Aunque, como decíamos, la razón de esta dualidad entre el código fuente –accesible a los humanos– y el programa compilado –necesario para su puesta en marcha en un ordenador– es estrictamente técnica; una de las consecuencias directas es que, efectivamente, el programa compilado se comporta de igual modo que un mensaje cifrado, de forma que no se puede leer y, por tanto, ni se puede comprender cómo funciona ni se puede modificar. Las empresas productoras de programas han utilizado esta característica para salvaguardar su propiedad intelectual, evitando el acceso al código fuente de dichos programas y manteniendo así secreto el modo en que estos están programados.

Richard Matthew Stallman (1953), programador y académico, denunció a principios de la década de 1980 que privar a los programadores del acceso al código fuente perjudicaba en gran medida la tarea de estos, además de suponer una ruptura de la solidaridad en el mundo de la ciencia, siempre acostumbrada a crear utilizando los avances del resto de la comunidad. En 1983, creó el proyecto GNU para desarrollar un nuevo sistema operativo que sería totalmente transparente a la comunidad, engendrando a la vez el movimiento para el software libre que culminaría, en 1985, en la creación de la Free Software Foundation.

Al hablar de Richard Stallman es imposible hacerse eco de las múltiples controversias que ha habido en cuanto a su persona y comportamientos y declaraciones que en el mejor de los casos deben considerarse tóxicos, y que culminaron en 2019 en su dimisión como presidente de la Free Software Foundation. No es nuestro objetivo defender dichos comportamientos, naturalmente, sino condenarlos. Pero ello no es óbice para reconocer la importancia de sus ideas en el campo del sofware libre.

1. El *hacking* y el ideario libertario del software libre

Un *hacker* –a diferencia del *cracker*, que podría ser asimilado, en gran medida, a un cibercriminal– es una persona que se dedica a crear y modificar programas, muchas veces por placer. El ideario del *hacker* es ciertamente parecido al del académico: documentarse, investigar, hacer un experimento, proponer el experimento a la comunidad, ser revisado por esta y, en el límite, ascender en la meritocracia sobre la base de los propios logros y de su contribución al crecimiento del conocimiento común.

Tomando esta ideología como base, Richard Stallman defiende que la ética *hacker* solamente es viable si el software cumple cuatro libertades:

- La libertad de usar el programa, con cualquier propósito (libertad 0).
- La libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades (libertad 1). El acceso al código fuente es una condición previa para esto.
- La libertad de distribuir copias, con lo que puedes ayudar a tu vecino (libertad 2).
- La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie (libertad 3). El acceso al código fuente es un requisito previo para esto.

Los programas que cumplen estas cuatro libertades (la libertad 0, la más elemental, junto con las otras tres) serán llamados programas libres o *free software*.

El sistema operativo GNU –iniciado dentro del proyecto GNU– será la punta de lanza del movimiento del software libre. Su evolución de un núcleo hacia un sistema operativo completo –a partir de la iniciativa del desarrollador Linus Torvalds– irá modificando el nombre original, GNU-Linux, hasta convertirse simplemente en Linux, como se conoce habitualmente.

Aunque en el ideario de Stallman el concepto de **libre** se refería, estrictamente, a **libertad** (en inglés, el término *free* es mucho más confuso que en castellano), una de las consecuencias directas de cumplir las cuatro libertades es que el programa se convierte, *de facto*, en un programa **gratuito**. Por otra parte, es tan cierto que el programa es gratuito como que su instalación, su reparación o su modificación para mejorarlo no tienen por qué serlo: del mismo modo que ocurre con los programas que no son libres, para modificar un programa libre hay que recurrir a programadores expertos que cobran por sus servicios. De hecho, hay un creciente sector a escala mundial que se está especializando en software libre y con unos resultados tanto o más lucrativos que los del sector del **software privativo**.

Otra cuestión a destacar de las cuatro libertades del software libre es que son mucho más estrictas de lo que a simple vista pueda parecer, especialmente la primera y la tercera. Tanto es así que, en 1998, surgió un sector en el seno del movimiento para el software libre que aboga por una mayor laxitud de las normas autoimpuestas, de forma que sea más fácil que más código pueda ser, si no liberado, sí abierto para ser estudiado y comprender cómo funciona y cómo realiza determinadas tareas. Una de las diferencias filosóficas fundamentales es si ese código puede ser o no incorporado a otros programas en los que el resto de código no es libre o, dicho de otro modo, si puede utilizarse determinado código en programas que requieren el pago de licencias para su utilización y que, además, suelen prohibir la copia y libre distribución de los mismos (primera y tercera libertades).

Dado que el origen del movimiento del software libre tenía ciertamente un interés académico –ver cómo funcionaban las cosas y poder aprender de ello–, se creó el **movimiento para el código abierto** (*open source*), que defiende, única y exclusivamente, que el código fuente sea accesible, pero sin el resto de condiciones del software libre. Existe una gran diferencia, tanto ideológica como técnica, entre el software libre y el software de código abierto.

Es esencial diferenciar, desde este momento, la gran separación tanto ideológica como técnica que hay entre el software libre y el software de código abierto.

En contraposición a esta laxitud del movimiento *open source* –aunque de hecho de origen muy anterior a su fundación–, el movimiento para el software libre comprendió, desde sus inicios, que era muy peligroso liberar código a una comunidad que no tenía por qué compartir el ideario del movimiento. Así, para evitar comportamientos de efecto polizón (también conocido por su denominación inglesa *free rider*), en el que alguien se beneficia del trabajo ajeno, las licencias de los programas libres –que contenían las cuatro libertades– incorporaron una cláusula que, en contraposición al *copyright*, se denominó ***copyleft***: cualquier programa libre con esta cláusula no solamente disfrutaría de las cuatro libertades prescriptivas, sino que obligaba a quien, haciendo uso de la segunda y cuarta libertad, modificase el programa, a distribuirlo exactamente bajo las mismas condiciones en que él accedió a dicho programa, a saber, con las cuatro libertades.

Dicho de otro modo: todo programa derivado de un programa libre con la cláusula *copyleft* debe ser, necesaria y obligatoriamente, también un programa libre.

La cláusula del *copyleft*, la dualidad del concepto *free* como libre y como gratis, y algunas características derivadas de las cuatro libertades que definen este tipo de software han desencadenado una serie de consecuencias o, mejor dicho, de

efectos que van mucho más allá de la ideología que promovió el software libre. O, a lo mejor, precisamente por estar fundamentado más en una filosofía que en aspectos meramente técnicos, el impacto del software libre ha trascendido, y mucho, el ámbito de lo meramente informático para entrar, de lleno, en el ámbito de la política y la economía hasta límites seguramente insospechados en 1983.

Antes de entrar en esta cuestión a fondo, eje central del próximo subapartado, cabe mencionar que existen en el mundo centenares de miles de proyectos de software libre que cuentan con millones de desarrolladores. Solamente SourceForge[1], uno de los principales –si no el más importante– repositorios y centros de desarrollo de software libre y abierto, tiene censados más de 400.000 proyectos llevados a cabo por más de un millón de usuarios.

(1)SourceForge: http://www.sourceforge.net

Hay que tener en cuenta que no todos estos desarrolladores son voluntarios –aunque sí la mayor parte–, sino que muchos son programadores profesionales que *liberan* el fruto de su trabajo *remunerado* a la comunidad (además de ofrecerlo a sus clientes, claro está), ya sea personal o institucionalmente a través de las empresas donde trabajan. En este sentido, muchas instituciones –un buen número de ellas públicas– deciden liberar el código y publicarlo bajo determinadas licencias y así ofrecer a la comunidad internacional o bien solamente dicho código –licenciando el programa con una licencia de código abierto–, o bien todo el programa, incluyendo las cuatro libertades con una licencia de software libre, como la **GPL**, promovida por la Free Software Foundation, la más común de ellas.

2. Política, economía y software libre

En un mundo con un sector TIC –y más concretamente, el sector del software– plenamente desarrollado y competitivo, y con una economía nacional o regional saneada y potente, muchas de las cosas que se dirán a continuación pierden bastante validez. Sin embargo, y muy a nuestro pesar, la mayor parte del mundo no cumple una o ninguna de las condiciones anteriores.

Aunque, con el paso del tiempo, se incrementan aquellos Estados que pueden entrar a competir eficientemente en el mercado del software, todavía Estados Unidos sigue acaparando el mercado mundial en lo que a producción de software doméstico y para la pequeña empresa se refiere –muy especialmente, la compañía Microsoft y sus productos estrella MS Windows, el sistema operativo, y MS Office, el paquete de ofimática, aunque también Apple, su competencia–, así como aplicaciones comerciales para la gran empresa, impulsadas desde IBM, Oracle o Adobe, por poner solamente unos ejemplos.

Un análisis rápido nos dará una situación aproximada del impacto directo de este hecho; la demanda nacional va a parar a productos extranjeros, con las siguientes consecuencias:

- Impacto negativo sobre la **balanza de pagos** por cuenta corriente.
- Impacto negativo sobre la **cotización de la divisa** nacional.
- **Sustitución de la producción nacional** o, más correcto en este caso, creación de **barreras a la entrada** en el mercado del software, por la amplia penetración del producto foráneo.
- **Dependencia tecnológica** del extranjero, ya que, por las razones apuntadas anteriormente, el software es un tipo de bien que, en su diseño, resulta difícil de copiar.
- «Fomento» de la **piratería**, al ser un bien muy fácil de copiar –en su forma final– y con un coste material muy inferior a su precio de compra.

Como podemos ver, el software privativo –como muchos otros bienes con fuerte dependencia respecto a la oferta extranjera, entre ellos, los combustibles fósiles– supone un verdadero quebradero de cabeza, a corto y medio plazo, para la inmensa mayoría de gobiernos por todos los impactos económicos que implica. Pero, a diferencia de muchos otros bienes –y aquí el paralelismo con los combustibles fósiles sigue siendo válido–, el software supone tanto un bien de consumo final como un bien de equipo, tal y como veíamos en el primer apartado al definir las características fundamentales de la sociedad del conocimiento, por lo que su importancia es tan estratégica que agrava, de por sí, el problema. En una sociedad neoliberal como la impulsada por la Organización Mundial de Comercio, el Fondo Monetario Internacional, la Unión Europea, la NAFTA o el MERCOSUR, las políticas de restricción de entrada de bienes –en

forma de cuotas, aranceles u otras vías– son inviables políticamente. Además, se añade la dificultad de sustituir las importaciones por el producto nacional, prácticamente inexistente. Por si fuera poco, se entra en un círculo vicioso de difícil solución: la presencia de software extranjero no deja crecer la industria local y la ausencia de dicho software extranjero –en el supuesto de poder prohibir o limitar su importación– dificulta el crecimiento de la economía por su alto componente estratégico.

Si, en lugar de tener un enfoque macroeconómico, lo hacemos en la economía doméstica o de una empresa, el análisis tampoco es muy esperanzador. En muchos países, y con más énfasis donde este software es percibido como estratégico para activar la economía de la región, las licencias de software privativo son prohibitivas; en muchos casos, incluso desproporcionadamente altas en relación con la renta media del país. Ante este problema, se presenta una disyuntiva: o bien se opta por no actualizar tan periódicamente como sería necesario el software, creándose una brecha digital por operar con programas –recordemos: capital– obsoletos, con menos funcionalidades y, en el fondo, mucho menos productivos, o bien se opta –como ocurre en prácticamente todo el mundo, con mayor o menor grado de implantación– por la **piratería**. Dejando al margen el efecto negativo que la piratería tiene ya no sobre determinada propiedad intelectual, sino sobre el concepto mismo de propiedad intelectual y de Estado de Derecho, socavando la observancia de la norma como forma de actuar en comunidad, el segundo gran efecto es que las denuncias –con sus respectivas sentencias a favor del demandante– por piratería pueden suponer, de la misma forma que la importación masiva de software, una fuga de capitales ingente, además del descrédito del país como políticamente estable y la consecuente disminución tanto de la inversión extranjera como del mismo comercio internacional.

Ante este panorama, tan pésimo tanto en el ámbito macroeconómico como en el microeconómico, el software libre –y, en ciertos casos, también el de código abierto– puede suponer la única vía de salida del atolladero. El software libre es, además de libre, gratuito. Y, como tal, basta con descargárselo e instalarlo para poder empezar a utilizarlo (libertad 0), se puede adaptar a las propias necesidades (libertad 1), se pueden hacer tantas copias como se desee y se puede distribuir por doquier (libertad 2). Si, además, abogamos por una solidaridad interterritorial, la libertad 3 nos ayuda en este cometido.

Si hasta aquí hemos visto los **efectos económicos**, los **efectos políticos** de operar con software libre son igualmente interesantes.

A pesar de estar viviendo una deriva hacia la forma de hacer economía neoliberal, el sector público sigue teniendo un gran peso en las economías nacionales, sobre todo a través de herramientas como la inversión pública y el gasto público, reducidas, a su mínima expresión, otras vías como la política fiscal y la política monetaria. Lo peculiar de las decisiones políticas ante un efecto económico exógeno a la Administración es que importa mucho el qué, el

cuánto y el cómo. No es indiferente que una administración gaste el dinero de sus contribuyentes en hacer un parque o una escuela, o en hacer una escuela en este barrio o en aquel otro.

Cuando una determinada Administración pública estatal decide proveerse de cierto software para sus decenas o cientos de miles de funcionarios, no resulta en ningún modo baladí que ese gasto –o inversión– vaya a parar a manos de una corporación extranjera o que, en cambio, vaya a fomentar la creación de un sector TIC de factura nacional, que no solamente no tendrá el impacto macroeconómico que antes apuntábamos, sino que generará riqueza interna, con su correspondiente creación de puestos de trabajo, efectos indirectos de dichos puestos de trabajo sobre el consumo agregado, y una larga serie de efectos secundarios en cadena que se extienden casi hasta el infinito. Dado que partimos del supuesto de que, en la mayoría de países, no existe un sector de las TIC en el ámbito de la programación, la decisión sobre **gasto o inversión pública** en el ámbito del software puede tener unas consecuencias muy positivas o muy negativas que, sencillamente, un gobernante no debería poder pasar por alto sin acabar rindiendo cuentas a sus electores y/o contribuyentes. Si, además, consideramos el sector de las TIC como la locomotora del desarrollo, las consecuencias de haber lanzado por la borda lo que algunos esgrimirán como una oportunidad de oro son ya inconmensurables.

En este momento, os habrá asaltado la siguiente duda: si el software libre es, además, gratuito, ¿cómo se combina este hecho con el gasto público en software libre? A esta pregunta podemos dar dos respuestas igualmente válidas. Por una parte, efectivamente, la gratuidad del software libre puede hacer que, simplemente, la partida que había que destinar a, por ejemplo, equipar escuelas con programas informáticos, se vea reducida drásticamente por la eliminación directa de las licencias de software. Y esos fondos liberados pueden destinarse a otras partidas deficitarias. Por otra parte, puede ocurrir que el software libre disponible se ajuste solamente en parte a nuestras necesidades y haga falta desarrollar nuevas especificidades y funciones para que nos resulte óptimo. Además del hecho de que las adaptaciones son muy difíciles (y a veces imposibles) en el software privativo –y las excepciones son carísimas–, es de suponer que, con el presupuesto ahorrado en licencias, una administración debería ser capaz de crear ese nuevo software contratando a los expertos nacionales del sector, entrando en el círculo virtuoso ya mencionado. Esto último tiene un corolario: todo lo que la Administración cree de nuevo, vendrá a sumarse al conocimiento del procomún, y toda línea de código nueva vendrá a añadirse al programa desarrollado hasta el momento, que podrá ser aprovechado por otras administraciones, ya sean extranjeras, ya sean nacionales, pero de otros niveles, o por las empresas del país, o por los usuarios domésticos del país.

Y con esta afirmación entramos en el terreno de la **adaptabilidad** del software libre. Aunque ya ha quedado prácticamente apuntado, el software libre siempre dejará abierta la posibilidad de adaptar o personalizar tanto su interfaz –la forma como se presenta al usuario– como las tareas que realiza hasta el último detalle. En la Administración electrónica, ello tiene dos beneficios cruciales:

- Podemos acercar a la **realidad cultural y lingüística** de los distintos administrados todo tipo de programa con el que queramos que aquellos se acerquen a la Administración a informarse o a realizar trámites administrativos.

- Podremos hacer que lo dispuesto en el **derecho administrativo** sobre procedimientos y documentos vaya a la par con lo que ejecuta un determinado programa informático, sin tener que prescindir de tal o cual procedimiento por ser incompatible con un programa privativo cerrado e invariable.

Otro aspecto que suele tener muy ocupado a los gobiernos de todo el mundo, más allá de la forma en que se acerca a la población, es cómo guarda sus secretos, es decir, la **seguridad**, y no sin razón. Veíamos en el apartado anterior que quien ostenta mayor riqueza de datos es quien proporciona mejor botín informático a los criminales. Pocas organizaciones disponen, como se podrá comprender, de más y mejores datos que la Administración. Si la Administración es, además, electrónica, suponemos que todos esos datos han pasado del papel a un formato digital, con lo que el riesgo no se puede desdeñar. Aunque el debate no está cerrado, existe una creciente comunidad que defiende que **el software libre es más seguro** que el software privativo. Entre los diversos argumentos que se barajan, dos son los que tienen más peso:

- Dado que el software libre –y en este caso, tanto este argumento sirve también para el software de código abierto– tiene el código accesible a cualquier persona, es fácil, si no inmediato, saber qué hace exactamente y con todo detalle cualquier programa que instalemos en nuestro ordenador o nuestros servidores. *Mutatis mutandis*, lo contrario ocurre con los programas privativos. Aunque no necesariamente todos y cada uno de los programas que instalemos tienen por qué tener algún componente troyano o de *spyware* –y, en la práctica, casi nunca sucede así–, la cuestión es que la gran cantidad de datos, así como la sensibilidad de algunos de ellos –como los datos de salud en la Administración sanitaria, o de protección de testigos en la Administración de Justicia–, son lo suficientemente importantes como para que cualquier precaución sea poca. Y volvemos al principio de la cuestión: ¿qué político se arriesga a perder los datos de sus ciudadanos?

- Por otra parte, y siguiendo la argumentación anterior, como el código es visible y utilizado por todos, son muchos más los usuarios que potencialmente detectarán un agujero de seguridad y, además, como el código se puede modificar y redistribuir libremente, se supone que el lapso de tiem-

po entre la detección del error de seguridad, su solución y la publicación de la misma será también mucho más pequeño.

Desde el punto de vista estrictamente técnico y macroeconómico, los expertos señalan un par de aspectos más a tener en cuenta. Dado que creemos que han quedado incluidos tácitamente en las explicaciones anteriores, los dejamos aquí apuntados brevemente:

- El **coste total de propiedad** del software libre se revela como menor que el del software privativo, tanto por las licencias y las actualizaciones como por el mantenimiento.

- La **independencia** tanto respecto de un vendedor como de una tecnología, en concreto, da mayor libertad al comprador, lo que a largo plazo también redunda en un menor coste, al hallarse ante una mayor competencia real, pudiendo escoger en todo momento proveedor y tecnología, sin crear lazos tecnológicos que puedan lastrar el desarrollo de sus sistemas informáticos.

Para concluir este apartado, debemos decir que la tendencia, en el ámbito de la Administración pública, va en el sentido de adoptar paulatinamente el software libre, con países abanderados como Brasil –alegando motivos tanto económicos como ideológicos– o importantes golpes de efecto como la sustitución de software privativo por software libre en las instalaciones de las agencias alemanas de seguridad nacional alegando, precisamente, motivos de seguridad.

Sin embargo, es honesto admitir también que las dificultades que cualquier iniciativa de implantación de software libre está encontrando son, en el mejor de los casos, un importante reto a superar. Por una parte, la gran penetración de los sistemas privativos, junto con la humana y natural resistencia al cambio, están haciendo que la población sea refractaria a cualquier alteración de su normalidad informática. Por otra parte, algunas aplicaciones de software libre distan de dar servicios parecidos –en algún caso, ni remotamente parecidos– a sus referentes en el software libre. Además, la todavía incipiente especialización en dicho software por parte del exiguo sector de las TIC de algunas regiones hace que el apoyo al usuario sea malo o inexistente, lo que agrava la angustia del usuario de estar perdiendo prestaciones a cambio de nada (recordemos que, mayoritariamente, pirateaba el programa).

No cabe duda de que, en estos momentos, los proyectos más exitosos son los pilotados por la Administración pública y con dos características muy marcadas:

- Impacto mínimo sobre el usuario final, por ejemplo, priorizando la sustitución de programas gestionados a través de una web, con lo que el interfaz no cambia.

- Impacto máximo sobre la visibilidad de las ventajas comparativas del software libre, por ejemplo, que todo el ahorro en licencias de software educativo irá a incrementar la compra de hardware para las escuelas.

Es, precisamente, este último aspecto sobre los costes de los bienes intangibles protegidos por la propiedad intelectual el que ha revivido un debate que, paradójicamente, es mucho más antiguo que el software. Así, el ideario alrededor del software libre y sus cuatro libertades se está viendo trasladado al ámbito de los contenidos digitales (texto escrito, grabaciones de audio y vídeo, etc.), que, como el software, son crecientemente utilizados en muchas actividades en la sociedad de la información.

Bibliografía

Jiménez Romera, C. (2002, junio). «Software libre y Administración pública» [en línea]. *Boletín CF+S* (n. 20). Madrid: Instituto Juan de Herrera. <http://habitat.aq.upm.es/boletin/n20/acjim.html>

Mas, J. (2005). *Software libre: técnicamente viable, económicamente sostenible y socialmente justo*. Barcelona: Gestión 2000.