La ética del software libre

26 Marzo 2012
https://www.genbeta.com/genbeta/la-etica-del-software-libre

Post invitado: Este artículo ha sido escrito por **Benjamí Villoslada**, informático que se define a sí mismo como "bitólogo y todoista". Es co-fundador de meneame.net y colaborador habitual de diversos medios, donde suele tratar temas relacionados con Internet y el software libre. Podéis encontrar más información sobre Benjamí en su blog personal y en su perfil de twitter.
Podéis leer la postura contraria redactada por Juan Quijano bajo el título Software Propietario y Open Source, formas de cambiar el mundo.

El **software libre** es consecuencia del método de la ética, que consiste en estudiar la bondad o la maldad sin interesarse en otros aspectos o enfoques. El análisis de lo bueno y lo malo no es caprichoso; exige proporcionar las razones por las que ciertas conductas son buenas y por lo tanto dignas de realizarse. También debe argumentar en contra de las malas conductas.

Si te molesta que digan que el software privativo es malo, no sigas leyendo. Recuerda: no tenemos en cuenta otros aspectos o enfoques al analizar la bondad o la maldad. No nos interesa si ese software privativo es más bonito, tiene más funcionalidades, es más popular, tiene mejor marketing, lo usa tu amigo, es el que conoces o lo que te da de comer. El análisis ético puede ser incómodo. A menudo estamos dispuestos a entregar libertad a cambio de comodidad, pero no es el caso de los impulsores del software libre. **Linus Torvalds** rechazó trabajar para Appley **Richard Stallman** habría sido camarero antes que programador de software privativo:

«La posibilidad más obvia era adaptarme a los cambios del mundo. Aceptar que las cosas eran diferentes y que yo debería abandonar esos principios y empezar a firmar acuerdos de no divulgación para sistemas operativos propietarios, muy probablemente escribir también software propietario. Así, me di cuenta de que podría divertirme escribiendo código y que podría ganar dinero —especialmente si lo hiciera en cualquier parte que no fuera el MIT—, pero al final, hubiera tenido que repasar mi carrera y decir «me he pasado la vida construyendo muros para dividir a la gente», estaría avergonzado de mi vida.

Así que busqué otra alternativa, y había una obvia. Podía dejar el sector del software y dedicarme a otra cosa. Bien, no tengo otras habilidades reseñables, pero estoy seguro de que podría haber llegado a ser camarero. [Risas del público]. No en un restaurante de lujo; no me contratarían, pero podría ser camarero en algún sitio.»

Para analizar la ética del software necesitamos conocer cuál es su importancia para la sociedad. Se trata de una herramienta práctica. El relato de una novela no significa ninguna limitación práctica para nuestras vidas, pero la cosa es muy diferente si se trata de una enciclopedia con entradas sesgadas. Aunque en ambos casos se trata de libros, lo importante es cuidar la transmisión y la evolución del conocimiento que determina lo que podemos saber y, por lo tanto, hacer.

Phillip G. Armour dijo que «No tratan el software como un medio, lo tratan como un producto, y este es el problema. El producto no es el software, el producto es el conocimiento que va en el software», y lo razonó distinguiendo cinco formas conocidas de almacenar conocimiento, analizando las características, ventajas y desventajas de cada uno de ellos. **Ricardo Galli** los resume así en su blog:

- **1. DNA**: Es el primer método de almacenamiento del conocimiento. El DNAexiste para almacenar el conocimiento de cómo crear vida, como una máquina de Turing. El conocimiento está profundamente empotrado, pasar de grado es obligatorio para la supervivencia de las especies. El conocimiento es persistente, pero se actualiza muy lentamente. No tenemos la capacidad de cambiar el conocimiento –todavía, o sí…– de forma intencionada. El DNA puede hacer crecer un objeto físico que interactúa y modifica el entorno.

- **2. Cerebro**: Es un «experimento» casi exclusivo de la raza humano: almacenar más conocimiento en el cerebro que lo que se hereda en el DNA. Usamos nuestro cerebro para almacenar el conocimiento que adquirimos, fue el segundo método de almacenar el conocimiento que conocimos. El conocimiento es muy volátil, pero podemos cambiarlo rápida e intencionalmente. Podemos aplicar ese conocimiento para afectar y modificar el mundo.

- **3. Máquinas y herramientas**: El valor más importante de una herramienta no es ella en sí misma, sino como ha sido creada y modificada. El conocimiento del creador de esas herramientas es lo que marca las diferencias. Se las suele llamar también "conocimiento sólido" y fue la tercera forma de almacenar el conocimiento. El conocimiento es bastante persistente, pero no es fácil de actualizar. Es intencional y existe para afectar el mundo exterior.

- **4. Libros**: Han permitido nuevas formas de depositar y acceder al conocimiento que hasta ese momento estaban confinados al cerebro. Hizo al conocimiento portable en el tiempo y en el espacio. El conocimiento es muy persistente, pero de actualización lenta. Aunque los libros son intencionales no tienen capacidad para cambiar al mundo.

- **5. Software**: Es la última forma conocida –de hace sólo unos 50 años– para almacenar el conocimiento. Después de unos inicios dubitativos, está creciendo a una velocidad vertiginosa. Multitud de personas están trabajando para obtener información de las fuentes más diversas, comprenderla, clasificarla y trasladarla a este medio, y entonces intentan validar todo ese conocimiento. Hay una razón para que se invierta tanto esfuerzo, este medio tiene las características que deseamos y que no tienen los otros medios: es intencional, persistente, de actualización sencilla y rápida, y sobre todo es activo.

Cuando usamos una herramienta del software de retoque fotográfico estamos aplicando conocimientos, elaborados a lo largo de milenios, sobre la naturaleza del ojo humano marcada por el DNA y el comportamiento de nuestro cerebro. Acciones que años atrás conseguíamos mediante máquinas y herramientas, que se documentaron en libros, pero que ahora están escritas en software. ¿Quién puede prohibirnos leer cómo funciona? ¿El motivo es porque describe los procedimientos en un lenguaje informático que se llama C y no en uno verbal que se llama español? **Que el lenguaje informático convierta el conocimiento en una herramienta debería ser una ventaja**. ¿Porqué no podemos copiarlo, que es tan fácil como pasar fuego, para compartir la herramienta y el conocimiento en forma de lenguaje informático? Sólo pueden prohibir todas estas cosas aquellos que heredaron el estilo de los monjes de monasterio, pero aquello fracasó porque no era bueno restringir el acceso y la distribución del conocimiento. Desde entonces, poco a poco, la humanidad fue más sabia. Y eso sí fue bueno.

El código

Es posible que alguien piense que tanto le da el código porque no sabe leerlo. No podrá modificarlo. Pero el desconocimiento no debería servir para restar importancia a libertades tan importantes como la segunda y cuarta –de entre las cuatro libertades esenciales del software libre:

- La libertad de usar el programa, con cualquier propósito.
- La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.
- La libertad de distribuir copias del programa, con lo cual puedes ayudar a tu prójimo.
- La libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie.

El hecho de no saber leer, generaciones atrás, ¿significaba que no era importante que algún vecino sí supiese? Fueron precisamente esos, los capacitados para leer, quienes luego hicieron que el conocimiento se extendiera porque lo compartieron. Hemos sido más sabios a medida que más gente sabía leer. **No es importante que uno no tenga previsto leer, lo importante es que nadie lo prohíba**. Si nos lo prohíben sabemos que estamos bajo un mal régimen, uno que

quiere ocultarnos algo para hacernos dependientes de vete a saber qué. Es la antítesis de la libertad. Un análisis ético nos dice que es malo.

Saber leer código hoy no es nada habitual, «cosa de informáticos». Pero quien lo entiende está en ventaja en un mundo donde el conocimiento se transmite en forma de bits a través de internet. En el Siglo XX era necesario saber inglés para publicar en los mejores medios; en el Siglo XXI es importante saber código para compartir conocimiento porque en el mundo digital no se trata sólo de texto, sino de texto más código que determinará el nivel de visibilidad. El papel y la tinta, rígidos, forman parte del pasado. ¿Porqué nadie discute la importancia de saber inglés pero duda de la necesidad de saber [pon el nombre del lenguaje informático que quieras]? La industria del software privativo nos hizo creer que es cosa suya. No es bueno porque significa una limitación. Hacen todo lo posible para que creamos que sin saber código podemos hacer muchas cosas. No son suficientes; saber código abre muchas más posibilidades porque podemos personalizar nuestros proyectos. No piensan en nosotros sino en ellos; quieren tener el control –basta leer las licencias de su software para comprenderlo.

Dicho todo esto en defensa del acceso y comprensión del código, posiblemente no necesitarás saber nada de código para llevar a cabo tus proyectos, pero tendrán más garantía en función a la cantidad de vecinos que lo dominen. Para que esto suceda el código debe estar disponible para ellos. Tiene que ser libre para que puedan satisfacer todas tus necesidades. De la misma manera que el mecánico del barrio, y no necesariamente el taller oficial, repara tu coche: puede hacerlo porque el motor tiene tornillos y existen manuales técnicos. Conocimiento completo, sin restricciones, garantiza el mejor soporte. ¿Quien quiere un coche sin capó ni tornillos en el motor?

El proceso creativo

El software libre no es un producto, sino un medio para transmitir y depurar el conocimiento. Nunca nos importó presentar algo muy básico porque era una forma de lanzar la idea y, si era buena, cautivar a otros para que la mejorasen. «Release early, release often» (presenta rápido, presenta a menudo) es uno de los principios de los autores de software libre. **El objetivo es satisfacer todas las necesidades de los usuarios lo antes posible y estimular a los programadores**. Hay proyectos que triunfan, otros no; depende del interés que despierten entre los desarrolladores. O interés por parte de usuarios dispuestos a pagar programadores para completar el proyecto, porque el software libre no es «cosa de ellos» sino de todos.

No siempre es necesario liberar el código porque iría contra una libertad básica: el derecho a la privacidad. Las licencias de software libre están pensadas para proteger a los usuarios mediante el acceso sin restricciones a todo el código que usan,

pero no tiene porque tenerlo gente que no lo use. Que no haya ninguna obligación a liberar el código ha hecho que algunos proyectos de gran calidad no estén completos para todo el público, y puede parecer que el software libre no evoluciona cuando no es así. Por ejemplo Google se reserva muchas mejoras de programas libres. Están en sus servidores, esto es, su casa, y tienen todo el derecho a la privacidad.

Además de los privados, existen miles de proyectos de software libre activos que son públicos. Las diferentes distribuciones de GNU/Linux (como Debian, Ubuntu o RedHat) los incluyen en sus ofertas de soluciones completas de sistemas operativos y utilidades de todo tipo. Se encargan de actualizarlos con las mejoras y parches de seguridad. Compiten por ofrecer las mejores soluciones y han conseguido que GNU/Linux sea cada día más fácil y amigable. La única novedad, para los usuarios de software privativo, que sólo podían optar entre Windows o MacOS, es que deberán escoger entre muchas distribuciones de GNU/Linux. Para algunos esto significa un problema, pero en realidad es algo que hacen cada vez que van al supermercado: compran productos «libres», como las verduras envasadas y congeladas (o no) en multitud de calidades y formatos. Cada fabricante hizo su selección y presentó su oferta. **Poder escoger es una forma de libertad porque podremos cambiar cuando no lo hagan bien**. Es bueno.

El trabajo de los usuarios es tan importante como el de los autores de distribuciones: si un programa no está completo, o falla, podemos contribuir a mejorarlo. No hace falta saber programar; a menudo basta con enviar sugerencias y avisar de los errores que da nuestra instalación o manera particular de usarlo. El resultado siempre será mejor software. Puede que la nueva versión no tarde mucho en aparecer –la distribución Ubuntu presenta dos nuevas versiones cada año y Debian cada día.

Existen limitaciones legales para el progreso de software libre, pero intentamos esquivarlas. Una es la falta de información de los fabricantes de hardware, que dificulta la tarea de programar controladores. Otra importante está en las patentes. Si tu distribución de GNU/Linux no puede reproducir DVD, algunos formatos de vídeo o según qué documentos, no sucede por limitaciones técnicas sino legales. El propietario de la patente sólo la concede bajo acuerdos que van contra la ética del software libre; haría falta aceptar licencias contrarias a la libertad. Las distribuciones (que a menudo tienen empresas detrás) no pueden arriesgarse a incluir código libre que viola patentes de software. Pero sí lo pueden hacer hackers individuales y buscando un poco encontrarás en algún lugar el reproductor de vídeos, DVD o documentos privativos.

El software libre no tiene coste económico por licencias ni copias porque no son cosas que tengan valor para los usuarios; no están dispuestos a pagar por ello. Cobrar por una copia no es incompatible con los principios del software. Por ejemplo, en Gimp podrían pedir 1000 euros por cada copia de su software de edición de imágenes, pero si nadie quiere

pagarlos no es culpa del modelo económico del software libre, sino de la realidad del mercado. Quizás los clientes estén dispuestos a pagar por servicios de soporte y adaptación del software. **Si la copia no tiene restricciones es más fácil que el programa se extienda y aparezcan más clientes**. Este modelo económico se está extendiendo con fuerza para la música, la literatura o el cine: no pagamos por la copia, que está disponible en internet, sino para que los autores puedan seguir trabajando. No son grandes cantidades, sino micropagos, porque miles pudimos bajarlo sin restricciones, hemos visto su trabajo, nos gusta, y creemos que es importante que la cosa siga.

La libertad de elegir

La industria del software privativo se defiende a medida que el software libre avanza. A menudo lo hacen con manipulación. Por ejemplo, cuando la administración lanza concursos para comprar servicios de software libre, alegan que esto va contra el derecho de elección. Hablemos de eso. He escrito este artículo en un portátil con Ubuntu y un sobremesa que funciona en Debian. En el caso del portátil me obligaron a pagar por un Windows 7 que no uso. El PC de escritorio incluyó un Vista que tampoco uso. Se trata de los equipos actuales, pero ya son 12 años usando GNU/Linux en los cuales he pagado por demasiados Windows que no quise. No tengo nada más que decir en cuanto al derecho de elección.

Conozco pocas personas que quisieron comprar un Windows; simplemente estaba allí, junto al hardware que escogieron en la tienda. Pero conozco a muchos que eligieron GNU/Linux y se quedaron. No todos hicieron un análisis ético ni están interesados en el código, pero es importante que sepan que este software apareció porque algunos hackers vieron que trabajar en software privativo iba contra sus principios éticos. Que lo que eligieron es bueno.

En Genbeta | Software Propietario y Open Source, formas de cambiar el mundo por Juan Quijano