Open Source Software: similitudes y diferencias con Open Data

Jens Hardings
Gerente adjunto de Spitec Ltda. (www.spitec.cl). Ingeniero Civil en Computación DCC Universidad de Chile; Doctor en Ciencias mención Computación, Universidad de Chile. Sus áreas de investigación han estado principalmente ligadas al FLOSS (Free / Libre / Open Source Software) o Código Abierto / Software Libre, Tecnologías de Información y Seguridad.
jens@hardings.cl

Lo primero que salta a la vista cuando se considera Open Data y la corriente de Open Source Software o Software Libre (en adelante le llamaré FLOSS para no perdernos en distinciones ético-filosóficas muy específicas), es que existe una filosofía en común. En ambos casos se busca poder reutilizar el esfuerzo para crear una obra intelectual, sea ésta software o datos y así lograr una mejor eficiencia en el uso de los recursos a nivel macro. Al mismo tiempo, se da acceso a todos, sin distinción del uso. Gracias a este acceso universal y a que no existen restricciones al uso y la copia de la obra intelectual o a obras derivadas de ella (esencialmente, modificaciones, o algún software/datos existentes a los cuales se agregan otros), existen muchas visiones diferentes y normalmente tiende a haber una convergencia en el uso, así como integraciones que se realizan con facilidad al procurar utilizar estándares accesibles a todos.

El tema del acceso universal puede sonar trivial, pero siempre existe la tentación de querer influir para bien en el mundo restringiendo el uso de software o de información a los fines "correctos". Sin embargo, cualquier restricción típicamente afecta más a quienes honestamente quieren utilizar el software o los datos para fines generalmente considerados como correctos, y no detienen demasiado a quienes tienen malas intenciones.

En cuanto a diferencias entre FLOSS y Open Data, podemos mencionar que en Open Data, al menos hasta el momento,

no existen estándares definidos, sino más bien principios o buenas prácticas. Y posiblemente no tiene demasiado sentido definir estándares muy estrictos respecto de modelos de datos y similares, dada la diversidad de modelamientos, realidades e interpretaciones. En ese caso, es útil considerar la dupla Open Data + FLOSS para poder procesar los datos provenientes de diversas fuentes de forma útil y productiva.

Otra diferencia sustancial es que las fuentes de datos en Open Data en general son gubernamentales o cuentan con algún tipo de financiamiento público. En cambio, si bien para el FLOSS esa línea se hace muy razonable, en la práctica los proyectos tienden a surgir de iniciativas privadas, y en muchos casos personales más que institucionales.

Una característica que parece ser propia de FLOSS más que de Open Data es el concepto de reciprocidad, cuya instancia más conocida es la cláusula de Copyleft en la licencia GPL. Esencialmente, ésta es una restricción que se impone a quien recibe una licencia para usar, modificar y redistribuir el software, de que no puede entregar o redistribuir ese software u obras derivadas de él bajo una modalidad de licenciamiento diferente a la cual él (o ella) recibió. O sea, si se redistribuye el software o algún software derivado de él bajo una licencia que no sea GPL, se pierde el derecho de uso original y con ello cualquier derecho de modificación, redistribución, etc. El objetivo de esta cláusula es perpetuar la libertad que entrega la GPL mediante el mecanismo de restringir las restricciones que se pueden imponer a un tercero, o imponer un "prohibido prohibir".

CONCEPTO DE AUTONOMÍA

Si bien una de las tentaciones al considerar el uso de FLOSS es asumir una ventaja de precio o más bien de costo, en la práctica esta ventaja, cuando efectivamente la hay, tiende a no ser tan relevante como se pensaría en primera instancia. Lo que es realmente una ventaja que resalta por sobre todas las demás, pero en general es subestimada, es tener mayor control sobre la plataforma tecnológica. El uso de estándares preferentemente abiertos y bien documentados es sólo el comienzo, porque dependiendo de la necesidad y de la envergadura del usuario, es posible realizar desde pequeñas adaptaciones, que se vuelven parte del proyecto original con lo cual la mantención futura no recae en el usuario, incluso es posible desde influir en el desarrollo futuro de un proyecto, hasta liderar ese desarrollo o uno alternativo si no hay consenso.

A ese mayor control le llamo autonomía, y es un símil al concepto de soberanía que existe a nivel de Estado. Mientras mayor autonomía o soberanía tenga un ente a través de tener el control de las herramientas esenciales que requiere para funcionar, menos le afectan desde decisiones externas -pasando por crisis de proveedores o socios estratégicos- hasta lisa y llanamente prácticas hostiles. Por otro lado, no tiene demasiado sentido intentar tener una autonomía del 100% tal como a un país no le conviene cerrar su economía para evitar incidencia extranjera y que no le afecten potenciales crisis mundiales. Un ejemplo concreto es el primer acercamiento que tuvo Venezuela al Software Libre, donde más que participar de una comunidad existente parecía que se buscaba realizar una comunidad completamente autónoma y separada al interior del país. Lo importante es lograr que el nivel de autonomía o dependencia de una empresa, tal como lo debiera ser el nivel de soberanía o dependencia económica de un país, sea consecuencia de una decisión y no un aspecto que se deje al azar. Por lo mismo, las decisiones de qué software utilizar en la gestión de la cual depende una organización, es una decisión estratégica importante porque genera un nivel de dependencia que por lo general no es considerado y que le corresponde a la alta dirección definir.

ROL DEL FLOSS HOY

Ya vimos que el FLOSS es un componente importante cuando una entidad busca aumentar su autonomía, y por sobre todo, es un referente en ese aspecto contra el cual poder comparar otras soluciones. Por lo mismo, es importante que se mantengan y se nutran los proyectos, para no perder esa oportunidad y ese referente. Basta recordar la época en la cual las prácticas comerciales de los dominadores del mercado obligaban a actualizaciones masivas solamente para poder seguir siendo compatibles con los demás, y el software libre ha jugado un rol fundamental en cambiar esa realidad.

Lo anterior se traduce en que el FLOSS actualmente cumple, aparte de los casos de nichos que no detallaremos aquí, al menos tres roles fundamentales en la industria TI:

1. Infraestructura base y commodity

La industria TI está muy enfocada en innovar a un ritmo muy acelerado, tanto así que quien no innova en esta industria no puede optar a ingresos interesantes. Como consecuencia, la infraestructura base y/o commodities son cada vez menos interesantes para la industria, que debe mostrar su valor agregado para seguir obteniendo ingresos y en algunos casos mantener una infraestructura base, con todos sus costos asociados, solamente para poder mantenerse en la pelea por las innovaciones de punta. De forma natural la infraestructura base y las aplicaciones que realmente son commodity debieran ser dominadas por soluciones FLOSS, ya sea porque un proyecto FLOSS se impone como el dominante en ese nicho, o porque un actor dominante en el mercado decide que le conviene dejar su solución disponible bajo un modelo FLOSS, que mantener su desarrollo bajo su exclusivo alero (y centro de costos), siendo por lo tanto candidato natural a permanecer ahora como solución FLOSS liderando ese nicho.

2. Cumplimiento de estándares: validación y "glue"

Este es un aspecto particularmente interesante considerando la temática Open Data, ya que el FLOSS siempre, y particularmente en todo el desarrollo de TCP/IP y todos los protocolos relacionados a Internet, ha sido muy apegado a estándares públicos y abiertos.

Las razones para que un proyecto FLOSS opte por utilizar estándares abiertos son bastante directas: en lugar de invertir esfuerzo en encontrar una solución a un problema, se opta por una solución existente que cumpla con cierto nivel de calidad y ojalá de validación, y tampoco existen razones estratégicas por las cuales realizar un desarrollo propio. Mucho menos hay justificaciones para mantener en secreto ciertas partes, dado que se diseña pensando en publicar todo el código. Por otra parte, muchos creadores de estándares realizan una implementación que luego publican bajo alguna licencia FLOSS, y sirve como referencia o incluso se puede incorporar tal cual en el software que los deba implementar.

En ambos casos, las implementaciones FLOSS sirven de validación y también, al ser un código disponible y modificable, sirven para adaptar un estándar a ciertas interpretaciones o traducirlos a otro estándar, actuando como el pegamento que junta dos sistemas normalmente incompatibles.

3. Herramienta comercial para clientes

Incluso para quienes no utilizan FLOSS, éste sigue teniendo alta relevancia como una herramienta comercial en dos aspectos:

1) Define el conjunto mínimo aceptable. Si debo pagar por una solución y perder autonomía, debe haber un valor agregado que justifique el sobrecosto frente a una alternativa FLOSS (con sus propios costos, pero también sus propias ventajas).
2) Presenta una alternativa real y usable a la cual acudir cuando hayan desacuerdos comerciales con proveedores TI; ya no es fácil para un proveedor estar en una posición de "mi solución es la única alternativa existente".

En base a lo anterior, queda claro que el rol del FLOSS es relevante y debe haber un interés por mantenerlo vivo. Eso no necesariamente implica financiar los proyectos, sino simplemente evitar romper el equilibrio del ecosistema en el cual funcionan los proyectos FLOSS.

Es importante recordar que en el FLOSS, los temas de autonomía, cumplimiento de estándares e interoperatividad son inherentes y los intereses de los creadores del software están perfectamente alineados con los de los usuarios, porque esencialmente tienden a ser las mismas personas. En cambio, en el software comercial, los proveedores siempre tienen el incentivo perverso de intentar crear diversos lock-in contra los cuales los clientes deben luchar. Por lo mismo, aunque hoy en día esos incentivos no se traducen en malas prácticas comerciales, es bastante lógico pensar que sin la existencia del FLOSS el escenario actual sería diferente. Por ende, el FLOSS cumple un rol similar a la milicia en los tiempos de paz.

CAMBIOS Y DESAFÍOS

Hoy en día, sobre todo en los proyectos exitosos, una preferencia por FLOSS no se justifica por grandes diferencias en costos. Los grandes cambios y desafíos tienen más relación con el potencial cambio en la forma en la cual la TI llega a los usuarios. Tiene mucho sentido que nos acerquemos cada vez más a un cobro por servicio en lugar de cobros de licencias que se basan en el derecho de autor que en realidad regula la copia, y sólo de forma indirecta mediante construcciones legales tiene incidencia sobre el uso ("si no lo usas de la forma que yo digo, te quito el derecho de haber hecho y/o utilizar la copia").

Por lo mismo, todos los conceptos ligados a entregar el software como servicio, tales como el cloud computing, generan un cambio en la forma de uso y es ahí donde se concentran los principales cambios y desafíos para la industria TI y el FLOSS naturalmente no escapa a ello.

Posiblemente también aumenten a futuro las discusiones sobre si el derecho de autor es efectivamente la mejor forma de promover el desarrollo de las TI, desde siempre han habido defensores de que un modelo más parecido al patentamiento sería mejor.

Infraestructura base

En esto el FLOSS tiene bastante que ganar. Cuando se habla que el 50% del poder computacional en el mercado es comprado por Google, Microsoft y Yahoo!, para todas ellas salvo una, el costo de licenciamiento por copia de software se vuelve un tema que crece exponencialmente. Así que, contrario a lo que se podría pensar, que el FLOSS es para las PYME que no tienen cómo financiar el par de licencias de software comercial que podrían ocupar, el tema de uso de FLOSS es mucho más relevante para las empresas que deben replicar su solución en miles o incluso millones de computadores que ofrecen sus servicios, en paralelo, pero donde necesitan una licencia por cada uno de ellos. Lo mismo sucede cuando uno mira qué sistemas utilizan los supercomputadores listados en el "Top 500", que en la mayoría de los casos utilizan FLOSS.

Neutralidad en el tratamiento de datos

No me refiero a neutralidad tecnológica, que es en sí mismo un oxímoron, sino neutralidad respecto del tratamiento de información. Hoy en día por ejemplo asumimos que existe neutralidad en la información que manejan y nos entregan los buscadores, aunque no tenemos herramientas para validar que ello efectivamente ocurra. Posiblemente en esa área exista la posibilidad de pensar en alternativas que sigan la motivación de FLOSS pero en relación a los servicios, en conjunto con Open Data, posiblemente con limitaciones de escala al menos por ahora. Sin embargo, ya existen ejemplos

concretos de iniciativas que van en esa línea: Wikipedia, OpenStreetMap y algunos proyectos de redes sociales.

FLOSS entregado como servicio

El caso de servicios que se basan en FLOSS, en particular FLOSS que utiliza alguna licencia que incorpora la retribución, así como la GPL, se genera un fenómeno interesante. En estricto rigor, el objetivo de la reciprocidad es evitar que alguien pueda tomar un software que es esfuerzo de una comunidad, hacer algunas modificaciones y luego adueñarse y/o lucrar con el resultado (una obra derivada en términos legales) sin entregarle esas modificaciones a nadie. Para no exagerar las limitaciones, y en parte porque también hay ciertas limitaciones prácticas, se dispuso que la reciprocidad entre en funcionamiento al momento de entregar el software a un tercero para que lo use. De esa forma no era necesario entregar modificaciones privadas, pero se resguardaba el acceso al código fuente para toda persona que usara el software.

Pero cuando para usar el software no es necesario tener una copia, sino por ejemplo acceder al software vía Web, cambia el modelo y ahora es posible tomar un software GPL, hacerle modificaciones y entregar un servicio mediante ese software modificado sin ninguna restricción de entregar esas modificaciones al menos a quienes utilizan el software.

Como respuesta legal a ese problema surgen licencias alternativas como la Affero, que hace más severa la cláusula de reciprocidad de la GPL al hacerla mandatoria al momento de ofrecer un servicio basado en un software bajo licencia Affero. De esta forma, un usuario de un software vía Web u otro mecanismo similar tiene el derecho de obtener una copia del código fuente, montar su propio servicio y/o verificar el funcionamiento del software.

En este ámbito aún no se llega a un mecanismo maduro y estable para manejar tanto la gestión del software mismo como la disponibilidad de los datos.

> Hoy en día, sobre todo en los proyectos exitosos, una preferencia por FLOSS no se justifica por grandes diferencias en costos. Los grandes cambios y desafíos tienen más relación con el potencial cambio en la forma en la cual la TI llega a los usuarios.

Eficiencia energética

Este es uno de esos temas de fácil solución técnica, pero que requiere una decisión muchas veces corporativa que puede ser más esquiva de lo razonable.

Contrario a lo que se podría pensar, la industria TI no es tan limpia como parece. Sin considerar la huella de carbono de la fabricación de los computadores sino tan sólo el consumo eléctrico, la huella de carbono de la operación de las TI hoy en día según varios especialistas supera la de la industria de aviación. Más aún en Chile, donde la energía eléctrica hoy es sinónimo de combustibles fósiles para su generación.

La eficiencia energética de un sistema requiere una alta coordinación entre el software y el hardware. A nivel macro (sistema operativo) no hay demasiadas diferencias, pero sí pueden haber diferencias muy notorias en el consumo que tiene un dispositivo, por ejemplo, una tarjeta de red o una de vídeo, cuando es utilizado mediante un driver privativo (o propietario si se prefiere llamar así) o mediante un driver Open Source creado con la poca información pública y obtenido mediante ingeniería reversa.

Independiente que lo consideremos justificado o no, el hecho objetivo es que hay mucha reticencia, en particular en la industria de tarjetas de vídeo, a entregar información detallada del funcionamiento del hardware. Esto puede llevar a que en algunos casos una diferencia entre un driver y otro pueda tener consecuencias más indirectas que las obvias de correcto funcionamiento y usabilidad.

CONCLUSIONES

El tema del FLOSS sigue tan vigente como antes, y tal como los temas relevantes y los desafíos de la industria TI van fluctuando (según muchos se van repitiendo pero con diferentes nombres y quizás diferentes énfasis), así también ocurre con el FLOSS. Hoy en día hay menos discusión absolutista, y más consideración racional y objetiva, lo cual a mi juicio es sano.

Asimismo, existe participación por parte de empresas de todo tipo y tamaño como parte de los ecosistemas que mantienen funcionando a los proyectos FLOSS, y con ello aumentan las herramientas disponibles para enfrentar los problemas y desafíos que tendrán estos proyectos a futuro. Incluso pienso que a futuro debiéramos ver en los concursos por fondos públicos, que disponer del software y/o datos generados durante la ejecución bajo parámetros de apertura (FLOSS, Open Data) tenga tanto o más connotación positiva como la actualmente exigida "protección de propiedad intelectual" de los resultados, que en la práctica significa restringir al beneficiario de los fondos la explotación comercial futura de los resultados. BITS