tecnimap 2002
www.tecnimap.com
VII JORNADAS SOBRE TECNOLOGÍAS DE LA INFORMACIÓN PARA LA MODERNIZACIÓN DE LAS ADMINISTRACIONES PÚBLICAS
A CORUÑA
15, 16, 17 y 18 de octubre de 2002
Palacio de Congresos y de la Ópera
Ponencias

SOFTWARE PÚBLICO LIBRE ¿HACIA EL COMPROMISO TOTAL?

Francesc Rocher

MODELO DE DESARROLLO DEL SOFTWARE LIBRE

Introducción

Cuando en 1.984 Richard M. Stallman fundó la Free Software Foundation poco se imagina él lo que el futuro le deparaba. El proyecto de la FSF nació con unos sólidos fundamentos orientados a la libertad para compartir el código fuente, modificarlo y redistribuirlo sin restricciones. Esta idea insólita, a primera vista, ha generado a largo plazo no sólo un conjunto destacable de software, sino que también ha establecido un sólido entramado de dependencias entre distintos proyectos que hace que éstos persistan en el tiempo.

El modelo

La mayoría de proyectos desarrollados como software libre, por no decir todos, destacan por compartir un conjunto de propiedades algo inusuales que les confiere un potencial mucho más poderoso de lo que a primera vista cabría esperar.

Presentamos a continuación las razones del éxito del modelo de desarrollo del software libre. Trataremos de poner de relieve, de una forma clara y sencilla, las claves principales que nos permitan entender su funcionamiento y su potencial para producir productos de calidad.

Ayuntamiento de A Coruña
XUNTA DE GALICIA

El modelo que planteamos a continuación parte de un conjunto de personas con un objetivo común. Para conseguir dicho objetivo se unen para cooperar unos con otros. Esto les supone un grado de satisfacción en proporción directa al grado de implicación. Lo contrario no es cierto, ya que también existen agentes no implicados directamente con el grupo que obtienen cierto grado de satisfacción.

Una vez establecido un grupo con cierta solidez, intentaremos mostrar que, aunque pueda llegar a disolverse, el resultado obtenido por éste puede perdurar en función del grado de complejidad obtenido.

En lo que sigue, cuando hablamos de proyectos de software libre nos estamos refiriendo a aquellos proyectos cuyo código fuente se distribuye bajo alguna de las licencias que permiten su libre distribución y modificación. Principalmente, aunque no de forma exclusiva, nos referimos a los proyectos distribuidos bajo licencia GPL .

La idea inicial

En principio podemos partir de una idea que alguien quiere poner en práctica o de una necesidad detectada. Sea ésta un programa, una librería o, en general, un software que cubra las necesidades de alguien. Esta idea inicial puede darse en numerosas situaciones y lugares: en una universidad, dentro un grupo de amigos que desean hacer algo concreto con sus ordenadores, una persona que desea mejorar o imitar los resultados obtenidos con otro software propietario. Se trata, pues, de un conjunto más o menos grande de desarrolladores con una idea común. Incluso a veces, inicialmente el grupo puede estar formado por una sola persona o puede estar geográficamente muy distribuido.

El primer paso suele consistir en anunciar la fundación del proyecto con el fin de captar a otras personas que aporten al grupo ideas para su planteamiento o implementación. O incluso para que colaboren con ellos en tareas tan diversas como la elaboración de documentos, traducciones, personalizaciones, pruebas o adaptaciones a otras plataformas. Esta fase puede tener mayor o menor éxito, en función del software que se desea producir. Por supuesto, el uso de Internet y de los múltiples recursos que en ella se encuentran son la base fundamental para este saque inicial.

Una vez que el grupo se ha lanzado al desarrollo, el proyecto suele pasar por distintas fases antes de producir una versión inicial, más o menos estable y usable. Pero lo que nos interesa destacar a partir de este punto es el modo en que el grupo coopera y porqué lo hace.

El equipo

Como ya hemos apuntado, partimos de un grupo de personas con una idea común: la producción de cierto software. El grupo en cuestión tiene dos características a destacar. Por una parte, excepto en ciertos proyectos que disponen de algún tipo de financiación (pública o privada), se trata de un grupo de personas geográficamente muy disperso. Por otra parte, la principal actividad de los individuos no es la producción del software en cuestión, sino el trabajo que cada uno de ellos realiza para mantener sus ingresos. Algún caso se dará, felizmente, en el que un individuo pueda permitirse el no trabajar y centrarse en el proyecto. Es posible, pues, que no se disponga en conjunto de mucho tiempo y de que, además, el tiempo disponible no se pueda sincronizar con el resto del grupo debido a las diferencias horarias.

Estos dos factores, combinados con un alto grado de motivación, producen un efecto en el grupo de máximo beneficio para el proyecto. Por una parte, los individuos tienden a no duplicar sus esfuerzos. Es decir, se dividen el trabajo de una forma consensuada y dentro de las capacidades de cada uno. Con esto se obtiene un primer beneficio: el avance en paralelo de los distintos módulos que luego formarán parte del proyecto final.

Este es un comportamiento racional que cabría esperar de cualquier sociedad organizada, pero lo destacamos aquí por su carácter altruista. Es decir, cada individuo colabora de este modo, cediendo su trabajo al resto de participantes, porque sabe que esto es lo mejor para el proyecto. Se trata de una forma de egoísmo ejercido mediante el altruismo: *"para obtener mi mejor provecho en el grupo daré al resto mi trabajo, y de esta forma el proyecto avanzará y mi satisfacción se verá realizada: dispondré de mi ansiado software con mayor rapidez y calidad".*

De la situación anterior se deriva automáticamente el segundo efecto positivo. Al existir un intercambio de información entre los participantes, los errores que éstos puedan detectar son rápidamente corregidos. Aparecen con esto dos tipos de interacciones a destacar. Por un lado, el avance en paralelo del proyecto: se tiende a añadir funcionalidad al producto concurrentemente. Y además, cualquier error detectado en la fase de desarrollo es rápidamente corregido.

Los individuos son conscientes que la no cooperación en las fases iniciales de desarrollo se traduce en un estancamiento del proyecto, en una bifurcación de éste o, simplemente, un abandono. Recordemos que el software que no necesita mantenimiento es aquel que no se usa. Luego se hace totalmente necesario que el grupo siga unido.

Distinguimos dos tipos de individuos en este escenario. Por una parte están los desarrolladores. Éstos forman el núcleo central del proyecto y suelen tener el *copyright* de éste. Son los *padres espirituales* del proyecto, quienes deciden publicar una nueva versión, incluir nuevas funcionalidades o rescribir el código con el propósito de mejorarlo. No se trata

de un conjunto cerrado de miembros. Al contrario, incluso a veces puede darse el caso que, con el tiempo, el núcleo de un proyecto no contenga ningún miembro inicial. Son pocos los casos, pero existen.

Por otra parte están los usuarios. Podríamos distinguir muchos tipos de usuarios según el grado de implicación de éstos en el proyecto. Para simplificar distinguiremos sólo dos tipos: los usuarios pasivos y los usuarios activos. Los primeros se limitan al uso y disfrute del software producido, mientras que los segundos se implican de una forma u otra en el proyecto. El grado de implicación abarca numerosas situaciones, que van desde la mera transmisión de *feedback* (reportando errores funcionales) hasta el envío de *parches*, pasando por la participación en listas de correo, creación de documentos y traducciones, etcétera.

En general, y para referirnos tanto a desarrolladores como a cualquier tipo de usuario, hablaremos de *participantes*. Todos ellos comparten el mismo objetivo: la consecución de un software determinado.

El sistema

No se ha de entender este modelo planteado como un sistema cerrado en el que no existe interacción con el exterior. Por sistema debemos entender el conjunto formado por los desarrolladores, los usuarios activos y el proyecto en sí. En la frontera del sistema se hallan los usuarios pasivos y el resto de proyectos desarrollados bajo el mismo modelo.

De nada serviría si el grupo central de desarrolladores no recibiese cierta cantidad de *feedback.* Es decir, se trata de los usuarios activos que no participan directamente en el desarrollo pero sí se interesan por su evolución y mantenimiento. Su actividad se centra en la detección de errores y posterior comunicación al grupo. Como antes, cuanto antes se detecta un error y se comunica al grupo, tanto antes se corrige y se alcanza la calidad deseada. De este modo, incluso los usuarios son conscientes de que cuanto mayor sea la calidad de *feedback* enviado, mayor será la calidad del código producido a posteriori, y, por tanto, mayor será el grado de satisfacción que éstos obtengan a largo plazo. Este mecanismo sirve también para que los usuarios con los conocimientos necesarios se unan al grupo, participando directamente en el desarrollo o enviando correcciones.

Así pues, se trata de un sistema que genera una determinada cantidad de información como consecuencia de las interacciones entre los participantes. Esta información se genera en el interior del sistema y se usa dentro de éste.

Pero el intercambio de información no termina en las fronteras del sistema. Todo programa, en mayor o menor grado, necesita integrarse con otros programas. En su expresión más básica, todo programa necesita ejecutarse sobre un

determinado sistema operativo (a excepción de un sistema operativo). Del mismo modo, todo programa necesita más o menos librerías, si no se quiere duplicar el esfuerzo de rescribirlas, claro está.

Se trata entonces de un sistema que intercambia información también con el entorno. Se relaciona hasta tal punto con ciertas partes de éste que se crean *dependencias funcionales:* para ejecutar el programa x se necesita el sistema operativo s y las librerías p y q. Luego, en función de lo crítico y complejo que sea el software, más fuertes son las dependencias creadas, con lo que un mayor número de individuos dependen indirectamente de éste y se implican en él. Esto es así por que, como hemos apuntado más arriba, se tiende a no duplicar esfuerzos. Por tanto, si una librería permite libremente la disposición inmediata de un conjunto de funcionalidades con un alto grado de calidad, ¿porqué iba alguien a rescribirla?

Hasta tal punto se crean tales dependencias que, llegada cierta masa crítica, el proyecto ya no tiene vuelta atrás. El sistema se hace cada vez más robusto y estable gracias al interés creado alrededor del software. Y al revés, cuantos más participantes, mayor es la calidad del software y más robusto es el sistema.

El resultado

Recordemos otra vez que el software que no se mantiene es aquél que no se usa. Visto de otro modo, y por la fuerte red de dependencias creadas entre distintos proyectos, el software que no se usa es aquél que no se necesita. Luego su abandono es inevitable.

Los proyectos más críticos y de mayor complejidad son los que mejor evolucionan, se adaptan a las necesidades y alcanzan mayores cotas de calidad. Son estos proyectos sobre los que recae un mayor número de dependencias, por razonesde estabilidad, usabilidad, confiabilidad y portabilidad. He aquí cómo el modelo de desarrollo del software libre es capaz de alcanzar cotas que a primera vista parecen inalcanzables, y de cómo sobreviven proyectos sumamente complejos como lo es el núcleo de Linux, XWINDOW, gcc y un largo etcétera.

Hay que matizar, eso sí, que se trata de una calidad que satisface al grupo y a los proyectos dependientes, entendida como *un estado del software producido aceptado en su mayoría por la comunidad de participantes*.

Dicho estado de calidad se refiere a las propiedades ya citadas de usabilidad, estabilidad, adaptación a las necesidades, portabilidad, etc. Estas cualidades, aunque buenas y deseables en todo proyecto y en concordancia con las definiciones típicas de calidad, no son todas las que se requieren para un proyecto.

La mayoría de los proyectos de libre distribución pecan de centrarse demasiado en el código producido. Y de hecho es así porque *debe* ser así, porque cada grupo de desarrollo se centra en una actividad que pocas veces tiene que ver con las definidas en los estándares de calidad. No queremos decir con esto que no se lleven a cabo actividades de test unitario, por citar un ejemplo, sino que nos referimos a actividades como la elaboración de planes de calidad o la gestión de requerimientos. Aunque esto no supone ningún problema, representa un leve inconveniente en cuanto a su uso y aplicación en las administraciones públicas.

SOFTWARE LIBRE EN LAS ADMINISTRACIONES PÚBLICAS

Una de las grandes ventajas de disponer del código fuente de un programa es el poder aprender su funcionamiento interno hasta el fondo, sin ningún tipo de restricciones. En principio no debería suponer ningún problema el que los usuarios dispongan del código fuente. Al contrario, esto es una ventaja respecto al software cerrado, ya que el acceso al código fuente permite que cada uno *rasque* allí donde le pica. Por ejemplo, aunque a primera vista parece contradictorio, los programas de encriptación de datos más seguros son aquellos cuyo código es abierto: la seguridad no reside en el algoritmo, si no en las claves usadas.

Compromiso con la sociedad

Dentro del escenario que hemos expuesto en el punto anterior, el hecho de disponer del código fuente es indispensable para la supervivencia del proyecto. Pero atención, se trata de un tipo de programas que los individuos usan sin ningún tipo de garantía. Las licencias de código abierto, y la GPL por excelencia, dejan muy claro que no hay ningún tipo de garantía para todo aquel que use el software en cuestión. Es opcional y posible, eso sí, que alguien decida ofrecer este servicio por un precio acordado entre ambas partes. De lo contrario, el usuario deberá correr con los costes que el uso del software le pueda ocasionar en concepto de pérdidas o daños.

En el modelo expuesto hemos partido de la base de que todo participante en un proyecto comparte el interés común por el software en sí. Y son conscientes del riesgo que corren al usar su propio software (o en el que han participado en cierto grado). Hay en cambio un matiz importante cuando este software se usa en la administración pública: su uso puede afectar a terceros que ni siquiera conocen la existencia de dicho software. Y como también puede afectar a las

personas que lo han desarrollado, estos dos factores podrían ser usados intencionadamente para obtener otro tipo de beneficio que el software en sí. Es decir, pueden aparecer intereses ajenos al proyecto en cuanto que producto software, de modo que la estrategia del egoísta puro empieza a tomar sentido.

El ejemplo más sencillo que se nos puede ocurrir: en un futuro el programa PADRE es de libre distribución. La gente puede participar libremente en su desarrollo y mantenimiento, explorar su funcionamiento y enviar correcciones, siguiendo el modelo abierto de desarrollo. Nada impediría, o al menos sería posible en teoría, que alguien cambiase intencionadamente algunas funcionalidades para sacar provecho de la situación. O peor aún, para perjudicar a otros.

Otro ejemplo: en un futuro se desarrolla un software para realizar determinadas gestiones en la administración local (ayuntamiento, por ejemplo). La persona encargada de dicha gestión sabe usar el programa y está al corriente que es de libre distribución y de que se puede descargar por Internet de cierto sitio. Nada impediría que esta persona (o en colaboración con otras) descargue el programa, lo modifique a su gusto y, una vez compilado adecuadamente, lo use en su puesto de trabajo para sacar provecho para sí o perjudicar a terceros.

Como se ve con este par de simples ejemplos, la responsabilidad del uso del software abierto en la administración pública afecta no sólo a una comunidad más o menos cerrada de participantes, sino que también afecta a las personas ajenas a él. Personas que no tienen los conocimientos técnicos necesarios para descargar el código fuente e investigar si éste se comporta como debiera. Que no pueden contratar los servicios de un programador o de una empresa para auditar el código. No se puede pretender, por tanto, que el mero hecho de usar software libre en las administraciones públicas sea un sello de garantía para la sociedad. Con esto no queremos poner en duda la competencia de quienes tienen a su cargo la instalación y explotación de los sistema de información en las administraciones públicas. Nuestra intención es invitar al lector y a reflexionar sobre estas cuestiones que, de buen seguro, le afectan de forma directa.

Es muy deseable que la sociedad se beneficie del uso y desarrollo del software libre, por supuesto y sin ningún tipo de dudas o reservas, pero hay que tener en cuenta que la sociedad la componen también personas sin los conocimientos para defenderse por sí solas ante los programas. Es imposible pretender que todas las personas alcancen dichos conocimientos, e incluso es no deseable.

Por esta razón entendemos que el uso del software libre en las administraciones públicas debe estar acompañado de cierto grado de responsabilidad, de tal forma que se garantice a la sociedad que nadie podrá usarlo a su favor o en contra de alguien. No se debería tratar de paternalismos gubernamentales para que la gente duerma tranquila pensando que alguien se ocupa a menor coste de sus problemas. Debería tratarse de una actitud de compromiso con el

software libre de modo tal que, mediante los mecanismos de arbitraje adecuados y de aseguramiento de la calidad, su uso repercuta en el máximo beneficio para la sociedad.

Estos ejemplos y razonamientos tampoco deben llevarnos a la conclusión de que la mejor solución es no usar el software libre. Al contrario, en un acto de madurez social y democrático, los programas de las administraciones públicas deberían ser abiertos y libres.

No dudamos en la capacidad de anticipación y reacción de quienes apuestan por el uso del software libre en la administración pública. Sirvan nuestras reflexiones para aumentar la confianza en él, para plantear nuevos modelos de desarrollo e incluso abrir nuevos mercados. Sabido es que el software libre no produce dinero, pero sí genera riqueza con los servicios que alrededor de éste se crean para su mantenimiento y evolución. Y con el ahorro de costes de licencia que supone.

Compromiso con el software libre

¿Qué significa estar comprometido con el software libre? En el modelo que hemos expuesto, significa, como mínimo, liderar un proyecto, participar activamente en el desarrollo de otros y, en general, adoptar una postura abierta y cooperativa con el resto del mundo.

¿Porqué las administraciones públicas deberían comprometerse con el software libre? En primer lugar, por el beneficio inmediato que supone el ahorro del coste de las licencias de software propietario. En segundo lugar, por que es un acto legítimo y democrático que las personas puedan colaborar en el desarrollo de herramientas usadas en la administración de ciertos aspectos de la sociedad. Y en tercer lugar, por que el buen estado de salud con que cuenta actualmente el software libre demuestra que el modelo de desarrollo abierto es robusto.

En otro orden de cosas, cabe destacar que el software libre y las administraciones públicas se asemejan en dos aspectos. En primer lugar, no existe ánimo de lucro en sus actividades. Y en segundo lugar, el factor tiempo no es determinante en sus actividades. Vayamos por partes. El proyecto GNU creado por la *Free Software Foundation* nunca se planteó que fuera rentable. Sólo importaba crear un software abierto que otros disponían como propietario, desarrollar y ampliar nuevas herramientas y, en definitiva, crear un entorno completo compatible con UNIX. Algo similar ocurre con las administraciones públicas: no se trata de empresas cuyo objetivo sea la maximización de los beneficios a la par que se minimizan los costes. Según el tipo de administración del que se trate, sus objetivos será unos u otros, pero siempre tienen a la sociedad en su punto de mira.

Lo mismo ocurre con el tiempo. El software libre avanza al ritmo que lo hace, no al ritmo que alguien le impone. Sus objetivos son a largo plazo, sin importar demasiado si se tarda un mes más o menos en tener disponible el producto. Y en cierto modo ocurre lo mismo con determinadas administraciones públicas: sus objetivos son a largo plazo, ya que de otro modo no los alcanzaría.

Pues bien, estos dos factores notablemente parecidos entre uno y otro, hacen perfectamente posible que las administraciones públicas se comprometan hasta el final con el software libre: la estrategia a seguir debería ser no lucrativa en cuanto a dinero (sí en cuanto a riqueza generada) y a largo plazo. Por supuesto que el beneficio inmediato del uso del software libre es el ahorro de licencias. Y de hecho ya hace tiempo que se están llevando a cabo acciones concretas en este sentido en todo el mundo. Pero esto no es suficiente. Se debería ir un paso más allá para que la sociedad se beneficie completamente del uso y d*esarrollo* del software libre: de su uso se beneficia a corto plazo, y de su desarrollo, a largo plazo.

CONCLUSIONES

Hasta este punto hemos puesto de relieve las claves del éxito del modelo de desarrollo del software libre. También hemos presentados algunos problemas e inconvenientes que se podrían derivar de su uso en las administraciones públicas, y cuál debería ser su grado de compromiso con el software libre para máxima garantía y beneficio de la sociedad.

Queremos presentar en nuestras conclusiones algunas propuestas concretas en relación al software libre usado, desarrollado o mantenido por y para las administraciones públicas, al que hemos querido llamar *software público libre*.

Nuestra propuesta se centra en la calidad del software. Pero en una calidad comprometida con la sociedad por los argumentos antes expuestos, lo que nos impulsa a pensar en algo más completo y complejo: la *Ingeniería del Software de Código Abierto*. El objetivo principal de ésta debería ser la calidad en los proyectos de tecnologías de la información desarrollados usando el modelo del software abierto. Entre sus muchos objetivos, cabría destacar la adaptación de tecnologías existentes al modelo de desarrollo de software libre.

Por parte de las administraciones públicas, y para desarrollar y apoyar al máximo el desarrollo del software libre, creemos que se debería hacer el esfuerzo de invertir en la creación de algún tipo de entidad, o de ampliar la actividad de alguna ya existente, con la finalidad de servir:

- Como punto de referencia para concentrar, liderar, arbitrar, organizar y distribuir las soluciones creadas siguiendo el modelo del software libre para las administraciones públicas.
- Como referente para el establecimiento de un nuevo tipo de mercado en el que participen también las empresas comprometidas con el software libre.
- Como plataforma para la adopción, integración, desarrollo o mantenimiento de herramientas para la gestión y desarrollo de proyectos basados en ingeniería del software de código abierto.
- Para la adopción y adaptación de nuevos estándares y parámetros de calidad para proyectos de software libre para las administraciones públicas.
- Como punto de información, difusión, formación, soporte y expansión del software libre.

La propuesta es ambiciosa, criticable, mejorable y nada sencilla. Pero de algún modo éste uno de los mejores momentos que ha vivido el software libre desde su creación en 1984. Cada vez despierta más y más interés en todos los ámbitos, y cada vez más son las empresas que invierten en él. Las administraciones públicas tampoco se están quedando atrás, tomando cada vez posturas más definidas y valientes. ¿Será éste el primer paso hacia el *compromiso total*? Queda mucho por ver, pero queda mucho más por hacer.