Enxeñería Sen Fronteiras Galicia
Migración a Software Libre de la Sede de Coruña
Documento I: Filosofía

Francisco J. Tsao Santín

Grupo de Sistemas de Información

13 de noviembre de 2008

Índice general

Capítulo 1

Breve introducción al software libre

"Omnis enim res, quae dando non deficit, dum habetur et non datur, nondum habetur, quomodo habenda est."

("Cuando una cosa no disminuye al compartirse con otros, no es bien poseída si sólamente se posee y no se comparte.")

– San Agustín, obispo de Hipona, *"De doctrina christiana"*, 397 D.C.

Si bien el software libre como movimiento es algo relativamente reciente (actualmente el proyecto GNU, el primer proyecto *"oficial"* de software libre, cumple ahora 25 años), lo que es la filosofía del software libre es tan antigua como el hombre mismo: su esencia está en crear y compartir conocimiento, para progreso de la humanidad. En la Edad Media, los ingenieros de puentes viajaban con sus diseños, investigaciones y observaciones de otras obras que no eran las suyas, y compartían este material con otros ingenieros.

Dando un salto milenario, la gran expansión del movimiento del software libre se produce en paralelo al crecimiento de Arpanet, red que unía centros militares en Estados Unidos, a la que posteriormente se agregaron las universidades y otros centros tecnológicos, convirtiéndose en Internet, y finalmente, llegando a todo ciudadano de a pie.

Para los no iniciados, habrá que explicar que esto del software es el conjunto de instrucciones que hacen que los ordenadores funcionen como que-

remos. Este software está escrito en un lenguaje determinado (el que sea, hay miles), que es *human-readable*, esto es, legible para seres humanos. Los primeros grandes avances de computación se produjeron en Estados Unidos y Gran Bretaña, así que es fácil entender por qué los lenguajes de programación utilizan bastantes términos ingleses.

Pero estos conjuntos de instrucciones, o código fuente, no los entienden las máquinas directamente. Para que así lo hagan, hay que hacer un preproceso (compilado-enlazado) que genera lo que se llama el código binario. Esto sí que lo entiende la máquina.

Si queremos que un programa funcione de forma diferente a como lo viene haciendo, tenemos, en general, que modificar el código fuente, y de nuevo, recompilarlo[1].

Durante décadas, código fuente de programas fluía por las redes (telemáticas, o del correo postal), entre los programadores. Pero a principios de los años 70, cuando aparecieron los primeros ordenadores personales, más rudimentarios como el Altair, o más modernos, como el primer Apple, esta filosofía cambió: un tal William Gates, director ejecutivo de una pequeña compañía llamada Micro-soft, con sede en un motel en Albunquerque[2], mandó una airada carta al Homebrew Club quejándose amárgamente de que el compilador de lenguaje Basic que ellos habían desarrollado andaba circulando por demasiados sitios sin que la empresa recibiera un dólar por su uso, que se iban a enterar todos, etc, etc.

Esta carta dirigida al primer gran club informático de la historia, es considerado como uno de los primeros referentes de aparición del software cerrado o *privativo*. ¿En qué consiste el concepto de software privativo? El desarrollador lo produce, y sólo permite el uso del código binario, bajo una serie de condiciones. Normalmente estas condiciones suelen referirse a una contrapartida monetaria, que varía según el uso, privado, docente, industrial, para uno o varias personas, para uno o varios puestos informáticos... aunque también pueden llegar a forzar exigencias del estilo de prohibir hablar mal del producto (como sucedía con las licencias de la base de datos Oracle). El programa no se *vende*, sino que se vende una licencia de uso, y, entre otras cosas, el código fuente suele estar en el cajón del desarrollador, para evitar

[1] evidentemente, se puede trabajar sobre el código binario, pero suele ser excesivamente complejo cuanto mayor es el programa, y los resultados pueden ser incontrolables, aun cuando el técnico tiene un nivel alto de conocimientos; tampoco es fácil invertir el proceso, *"desensamblar"* el programa

[2] lugar conocido en Estados Unidos especialmente por la actividad del ganado vacuno

que el programa sea analizado y modificado. Se vende, en definitiva, el uso de una *"caja negra"*, que además de hacer lo que se supone que debe hacer, puede estar haciendo otras cosas menos amigables para el usuario.

Pasan los años, y este movimiento de codicia llega al laboratorio de Inteligencia Artificial del Instituto Tecnológico de Massachussets. En él se había vivido durante los 60 y 70 un tiempo de esplendor de la *cultura hacker*[3], pero poco a poco un hacker que respondía al nombre de Richard M. Stallman (más conocido como RMS) se había quedado solo. Como muchas revoluciones, ésta empezó con un accidente: una avería en una impresora, causada por la mala programación del controlador (el controlador es el programa que gestiona el funcionamiento del dispositivo). RMS se comunicó con la gente de Xerox, explicándoles que si le enviaban el código fuente del controlador, el sabría arreglarlo. Pero la respuesta que tuvo de Xerox fue la negativa, y sólo después de muchas negociaciones, accedieron a dejarle ver el código fuente del programa, pero con unas condiciones legales que hacían absolutamente imposible el solucionar el problema.

Como persona de carácter, RMS entró en cólera, rechazó definitivamente la propuesta de solución de Xerox, y tomó la decisión más inaudita de todas: crear su propio sistema operativo, distribuído bajo unas reglas que evitaran situaciones tan surrealistas como esta. De nuevo, para los no iniciados, un sistema operativo es el conjunto de programas que permiten funcionar a las aplicaciones que utiliza el usuario final. El elemento fundamental del sistema operativo es el núcleo o kernel, que es el programa que gestiona los recursos de la máquina, el que asigna tanta memoria o tiempo de proceso a una determinada aplicación.

Para proteger el trabajo que se iba a acometer, RMS y la gente que se fue uniendo a él en el proyecto GNU, se dio un soporte legal al software desarrollado, en forma de licencia. Esta licencia fue la General Public License (GPL), que hoy ya ha alcanzado su tercera versión. La GPL y otras licencias similares establecen las cuatro libertades:

- Libertad 0: libertad para usar el programa sin ninguna restricción
- Libertad 1: libertad para analizar y modificar el programa
- Libertad 2: libertad para redistribuir el programa

[3]entendiéndose ésta como la cultura de la experimentación, del compartir información, del llegar hasta los límites de la máquina

- Libertad 3: libertad para redistribuir el programa modificado

En concreto, la GPL pertenece a lo que se conoce como el conjunto de licencias de *copyleft fuerte*, esto es, incluye una cláusula que obliga al que redistribuye el programa, que lo haga bajo las mismas condiciones en que las recibió. Esto permite que se propague la libertad de software, e invita al resto de la comunidad tecnológica a colaborar.

En los años siguientes, a lo largo de la década de los 80, el proyecto GNU (acrónimo recursivo que quiere decir GNU is Not Unix), fue creciendo en desarrolladores, herramientas y usuarios. Primero se programaron las herramientas básicas para construir un sistema: editor de texto (Emacs), compilador de lenguaje C (gcc), depurador de código (gdb) y asistente de compilación (GNU Make). Después vino todo lo demás.

Pero a principios de los 90 faltaba el elemento fundamental para hacer el sistema autosostenible: el núcleo. Para GNU se había apostado una arquitectura de núcleo innovadora en aquel momento, pero muy difícil de programar y, sobretodo, de depurar. Y aquí la historia añade dos escenarios: California y Finlandia.

En la Universidad Berkeley se había venido desarrollando desde finales de los 70 una variante del sistema operativo Unix, conocida como BSD (Berkeley Software Distribution). Tan importante era, que sobre ella se implementaron por primera vez los protocolos base que rigen hoy Internet. A finales de los 80, varios seguidores de la filosofía del software libre empezaron a pelear porque el sistema operativo se liberara, pero se encontraron con la oposición frontal de determinadas empresas, llegando el caso a los tribunales, y paralizando el desarrollo de BSD.

Por otra parte, en esos mismos años, un estudiante de la Universidad de Helsinki llamado Linus Torvalds, se pelea por algo más sencillo, como el poder hacer las prácticas en el ordenador de su casa. Comprar un Unix comercial era algo absolutamente prohibitivo, y la otra posibilidad que tenía a mano era un *Unix de juguete* llamado Minix. Aburrido, Torvalds decidió un día hacerse su propio núcleo, utilizando para el desarrollo herramientas GNU. Y publicó una primera y muy rudimentaria versión, que tuvo una acogida entusiasta entre un montón de desarrolladores de software de un canal de news. El momento era crítico: Internet estaba dejando de ser algo reservado a militares y algunos centros americanos de investigación, para llegar a todo

el mundo[4]. En un año, lo que primero Torvalds llamó Freaks y que alguien renombró como Linux, ya tenía una primera versión estable.

Hombre rico, hombre pobre. Mientras que un sistema muy completo como BSD bregaba en el complejo sistema jurídico estadounidense, Linux ganaba adeptos entre estudiantes y profesionales. En 1994 la sentencia fue favorable a la gente que litigaba a favor de la liberación de BSD, teniendo que eliminar menos de un 5 % de las líneas de código. Pero el tiempo de estancamiento había sido crucial para que se impusiera el kernel Linux en el mundo del software libre. Aún así, hoy en día, de aquel 4.4BSD surgieron varios proyectos de importancia, los más destacables FreeBSD, usado en ISP's y en multinacionales como Yahoo en servidores críticos, o OpenBSD, que por su enfoque a seguridad tuvo una beca durante años de la NSA (Agencia de Seguridad Nacional de EEUU), hasta que su director de desarrollo, Theo de Raat, se enfrentó a la administración americana por la invasión de Irak en la Segunda Guerra del Golfo.

Los años 90 supusieron la consolidación del conjunto GNU/Linux en diversos lotes o **distribuciones**. Las distribuciones surgieron para facilitar la vida a los usuarios, tanto a la hora de instalar como de administrar y actualizar los sistemas.

La primera gran distribución conocida fue Slackware, creada por Patrick Volkerding. Era muy sencilla en su concepción, y ha tardado años en cubrir uno de los objetivos de la distribución (la facilidad de actualización), pero ha llegado hasta nuestros días. Después, vinieron las dos grandes distribuciones que todavía hoy marcan la pauta en el software libre: Debian, proyecto fundado por Ian Murdock, y RedHat, distribución creada por Mark Ewing. Ambos proyectos tienen su propio sistema de paquetes, esto es, la parte de la distribución que controla la información de cada paquete (versión, dependencias, etc).

Con la mejora en la facilidad de instalación y actualización de las distribuciones, el uso de GNU/Linux fue creciendo de forma exponencial. No sólo entre hackers, estudiantes y aficionados: empresas y gobiernos se dieron cuenta enseguida del valor y la calidad de las herramientas libres y las fueron

[4]En el año 93 los únicos ordenadores de la Facultad de Informática de la Universidade da Coruña que tenían acceso a Internet eran 8 o 10 equipos reservados para investigadores y la máquina de la Comisión de Extensión Universitaria, donde un puñado de hackers recogía información relevante y la dejaba disponible para que estudiantes y profesores la descargaran por red interna

incorporando a sus procesos industriales. En los primeros años del nuevo siglo, se ha llegado al dominio absoluto en servidores en Internet, en muy buena parte gracias a la popularidad del servidor web libre Apache, además de barrer en el Top 500 de supercomputadores del mundo, de ir al espacio en satélites de la NASA y de la Agencia Espacial Europea, de incorporarse en miles de sistemas empotrados (routers, PDA's, móviles...).

En este escenario, y a día de hoy, el único campo que le queda por copar al software libre es el del usuario final. Si bien es cierto que los entornos gráficos más avanzados superan con creces en usabilidad y espectacularidad a los de cualquier software privativo, los sistemas operativos libres tienen varios escollos que superar todavía:

- la falta de soporte para varias piezas de hardware, ya que muchos fabricantes se niegan a hacer públicas las especificaciones de sus productos, lo que obliga a los desarrolladores a realizar controladores para estos dispositivos por el difícil trabajo de la ingeniería inversa. [5]
- el estado embrionario de aplicaciones para campos específicos, como por ejemplo, CAD
- la falta de formación informática del usuario final, poco preparado para resolver problemas
- la actitud del usuario final ante el cambio, desde el temor hasta la desidia

Y explicado el contexto actual del software libre, vamos a ver que pasa con la ingeniería...

[5] Un caso paradigmático ha sido el de los chips Atheros que se encuentran en una gran cantidad de dispositivos inalámbricos. La compañía se negaba a ofrecer especificaciones y controladores libres, aduciendo problemas legales. Cuando el esfuerzo de varios años por parte de desarrolladores de Linux y OpenBSD estaba a punto de conseguir un controlador absolutamente libre, la compañía los contrató y en unos meses liberaron un controlador libre:*para este viaje no necesitábamos estas alforjas.*

Capítulo 2

El software libre y la ingeniería

Indudablemente, la ingeniería se ha visto afectada por la popularización de los computadores personales. Hace no más de 30 años, los cálculos estructurales se hacían con regla de cálculo o con modelos muy simplificados en los primeros (y prohibitivos para la economía de las pequeñas empresas) ordenadores personales. Los estudiantes aprendían a mejorar la presencia de sus dibujos con tiralíneas que hacían trazos con tinta china, un proceso desagradable que con la llegada de los grafos mejoró algo (era desesperante tener un dibujo completo que en último momento se echaba a perder porque una gota de más de tinta se perdía bajo el bisel de una regla). Los colores se ponían con lápiz o algunos tipos de pinturas que homogeneizaban la mancha.

Hoy calculamos con poderosos programas de estructuras, sobre unos ordenadores que están en cada hogar. Si con eso no llegara, tenemos supercomputadores a precios de risa, lo que permite refinar los cálculos hasta extremos que provocan risas histéricas en los supervivientes de aquel equipo de 300 personas que calculó a mano la bóveda del Mercado de Algeciras en los años 50, a las órdenes de Don Eduardo Torroja. Por si fuera poco, estos programas de forma casi automática generan hasta los planos del diseño, planos que, gracias a Internet, vuelan de un punto a otro del planeta a velocidades cercanas a la de la luz.

En definitiva, el uso de las TIC's es imparable en la ingeniería. Y sin embargo, por ser una revolución tan reciente, se han importado a ellas, de manera antinatural, una serie de tópicos de la economía tradicional, que amenazan con provocar desagradables frenazos en este movimiento con tanta inercia.

Pongamos como ejemplo un suceso acaecido hace algunos años. La com-

pañía de software Autodesk, productores del programa de CAD multipropósito AutoCAD, envía una misiva al Colegio de Arquitectos de Madrid. En esta carta, aparece la estampa de un hombre tras unas rejas, y el pie de foto reza *"Tu serás el siguiente"*. Este hecho, insólito de por sí por las formas, refleja una realidad: la dependencia de la ingeniería y arquitectura de un programa informático y un soporte documental, propiedades de una determinada compañía.

La reacción del Colegio de Arquitectos fue la de encargar a otra compañía el desarrollo de un programa de CAD a mucho menor precio que el que impone Autodesk a sus productos, de tal manera que se ofreció a sus colegiados costes por licencia un orden de magnitud menor que el del precio de AutoCAD. Esto no solucionó el problema: por un lado, porque la acción de muchos profesionales fue la de comprar BrisCAD para disponer de una licencia legítima ante una inspección, y seguir trabajando *bajo la alfombra* con AutoCAD. Por otra parte, el trabajo de la compañía que desarrolla BrisCAD y cualquier otra con intenciones similares (proporcionar un CAD que permita a los profesionales trabajar con archivos dwg), está condenado al fracaso constante, al ser vagones de cola de desarrollo del formato que Autodesk modifica en cada nueva versión o actualización de AutoCAD. El mal llamado *standard de facto* evoluciona a gusto de sus creadores hasta el punto de incluir en sus últimas versiones un sistema de DRM lógico (Digital Rights Management) que permite a Autodesk bloquear la apertura de archivos .dwg si no están generados con un AutoCAD legítimo, si así lo desean.

Ha habido reacciones más o menos coordinadas ante esto. Una de las más significativas es la de la Open Design Alliance, un consorcio cuyo lema es el de *devolver la propiedad de los diseños*, actualmente en manos de Autodesk, a sus creadores. La idea es buena, pero la ejecución es bastante deficiente: la ODA ofrece en exclusiva a sus socios programas o librerías de programas que pretenden ser interoperables con las versiones de AutoCAD que van saliendo al mercado, además de para otros formatos como el .dgn que usa Bentley en su producto estrella, Microstation. Bentley forma parte de la ODA, y ha anunciado a bombo y platillo su iniciativa OpenDGN, que prometía a desarrolladores las especificaciones de su formato. Nada más lejos de la realidad, el buzón de Bentley a este respecto parece un pozo sin fondo ni respuesta. DGN V8, al contrario de su predecesor DGN V7, adolece de los mismos problemas de interoperabilidad que DWG.

Echando un vistazo al mundo industrial *ordinario*, observamos la progresiva estandarización de muchos productos y procesos. A ningún fabricante de

armaduras de acero se le ocurre fabricar una de dimensiones aleatorias, sino que se ciñe a los standards que publican organismos neutrales (neutrales al menos sobre el papel, vaya). Las empresas adoptan procesos de fabricación que cumplen determinadas normas ISO, de tal manera que sea fácil para las terceras personas saber que grado de calidad alcanzan. Esta idea, después de muchos años de otro conjunto de *formatos de facto*, se llevó al mundo de la ofimática, de la mano de desarrolladores de software libre. Una iniciativa, la Oasis Foundation, en conjunción con expertos de todo el mundo, diseño unos standards de formatos de fichero para ofimática, Open Document Format, que se han convertido en *standard real* a través de ISO. Esto ha permitido a programadores de cualquier suite ofimática implementar un formato de documentación que realmente pudiera interoperar entre todas ellas.

Esto permitió a administraciones de todo el mundo, realizar comunicaciones con sus ciudadanos sin la obligación para éstos de poseer determinados productos de una empresa. De forma tímida al principio, pero cada vez más, las administraciones están adoptando este formato junto con otro standard, el archiconocido pdf, además de los que ya nacieron como tales, los standards del W3C (Wide World Web Consortium).

La reacción de determinada compañía con sede en Redmon no se hizo esperar: ante la perspectiva de perder un mercado tan importante, por un lado *se habló* de presiones a los gobiernos aprovechando su posición de casi monopolio; por otra parte, documentaron su propio standard, el OOXML, en realidad un subconjunto de los formatos de la suite MS Office, que no sólo no cumple las normas básicas de los standards (dependencia de programas cerrados y de patentes no liberadas a todo el mundo), sino que además su reciente aprobación por ISO está en entredicho por irregularidades en las votaciones de los comités nacionales que tenían que decidir sobre ello. Pese a todo, parece que el avance de Open Document Format es imparable.

Este ejemplo es una referencia en aplicaciones de propósito general. Y se espera, y se desea por muchos, que en el futuro se vaya extendiendo a sectores más pequeños. No olvidemos, que lo que ha hecho grande a la ingeniería, ha sido el libre fluir de conocimiento.

Capítulo 3

El software libre y a cooperación para el desarrollo

En los puntos anteriores se ha intentado hacer énfasis en la libertad de conocimiento y la interoperabilidad como condiciones para un avance rápido en la ciencia y la tecnología. Esto, indudablemente sucede en el mundo occidental. Pero, ¿qué pasa en los países en vías de desarrollo? Si para nosotros la libertad de conocimiento es importante, para ellos es esencial. Si para nosotros la interoperabilidad entre tecnologías nos permite avanzar más rápido por la sinergia, para ellos supone la misma sostenibilidad de su progreso.

La ingeniería es eminentemente práctica. Las soluciones deben llevar a la buena resolución de cada problema. Este planteamiento es el que ha llevado a pasar por encima de la legalidad vigente y no pagar por el uso de tecnología privativa. En los países en vías de desarrollo, es algo que sucede con más naturalidad que en el mundo occidental. Económicamente, no deja de ser un importante volumen de economía sumergida. En muchos lugares, eso tiene poca importancia, porque por muy contundentes quieran ser las empresas con sus acciones legales, de donde no hay, no se puede sacar. Tampoco les interesa.

Porque lo que sí les interesa a las empresas de software privativo es generar *masas de yonkies* dependientes de sus productos, para que en el momento en que sí puedan pagar, forzarlos a ello, aprovechando esa dependencia generada de los desarrollos realizados sobre esa tecnología, haciendo retroceder económicamente a empresas y administraciones.

Caso grave entonces es el que los agentes de cooperación para el desarrol-

lo se conviertan en *camellos* que *regalan unas papelinas* para generar dependencia. Evidentemente, entre generar dependencia y economía sumergida, y salvar vidas humanas, lo prioritario es salvar vidas humanas. Pero si, como cada vez más, hay alternativas libres para muchas de las herramientas usadas en cooperación para el desarrollo, aún cuando puedan estar menos pulidas, es una obligación ética de OG's y ONG's hacer el esfuerzo de la adaptación y aprendizaje de las nuevas tecnologías libres, que por definición, incrementan conocimiento y son sostenibles económicamente, lo que, conceptualmente, las introduce en el saco de TpDH (Tecnologías para el Desarrollo Humano). Por ello, y pese a los pequeños dolores de cabeza, las incomididades, el esfuerzo del reciclaje, y alguna que otra situación límite, ESF Galicia está apostando en y desde su sede en Coruña por la migración, en la medida de lo posible (que es bastante), a software libre, en concreto a su variedad GNU/Linux como sistema operativo, y OpenOffice como implementación de referencia de suite ofimática de formato abierto.