UNIVERSIDAD REY JUAN CARLOS

Máster en Sistemas Telemáticos e Informáticos

Escuela Superior de Ciencias Experimentales y Tecnología

Curso académico 2010-2011

Apuntes Software Libre

Índice general

Índice de figuras

Capítulo 1

Introducción al *Software Libre*

1.1. Introducción y Motivación al *Software Libre*

Navegadores *web*, gestores de correo electrónico, juegos, herramientas para edición de vídeo, reproductores multimedia, suites ofimáticas, aplicaciones de mensajería instantánea, analizadores de tráfico, servidores *HTTP*, ...

Actualmente el *software libre* nos ofrece todo tipo de alternativas tanto en el segmento de servidores como en aplicaciones de escritorio que estamos acostumbrados a usar a diario. Pero, ¿qué es el *software libre*? ¿Cuándo nació? ¿Cómo ha evolucionado? A estas y a algunas otras preguntas intentaremos responder a lo largo de este primer capítulo de la asignatura.

1.1.1. El *Software Libre* como Alternativa

El modelo tradicional de producción ha conllevado a ciertas anomalías, en el fondo, todas provenientes de la idea de que tan sólo el productor puede mejorar su producto.

Por otro lado, el productor impone ciertas condiciones a través de la licencia del *software*, licencia que se deberá aceptar con tal de poder hacer uso del programa, hasta el punto de que en ocasiones, en el momento de realizar la compra estamos aceptando los términos de la licencia sin ser conscientes de ello.

Otra idea heredada a lo largo de los años es que tan sólo se podrá hacer negocio (ingresos y ventas) si se tiene el control absoluto sobre la copia y redistribución. Se trata al mismo tiempo de un modelo de negocio basado en la exclusividad, de manera que un único productor ofrece ese programa, produciéndose por tanto monopolios (sabido es el ejemplo de *Microsoft Office*, empleado mayoritariamente de una manera muy probable por tradición y por desconocimiento de alternativas). De este modo, se produce un monopolio no ya de producto, sino que llega a asociarse ese determinado producto con la empresa que lo produce. Además, el productor será el único con la potestad para resolver problemas del *software* y mejorarlo, con lo que se frena también el proceso de mejora e innovación.

Así pues, el *software* libre supone, en cierto modo, un punto de ruptura con respecto a esta visión tradicional dadas las características que presenta y que estudiaremos a continuación.

1.1.2. ¿Qué es el *Software Libre*?

En el espectro del mundo del *software*, se puede afirmar que *software* libre y *software* privativo son antagonistas, extremos contrarios el uno respecto al otro.

Al contrario de lo visto en la sección anterior para el caso del modelo tradicional, el *software* libre presta una atención muy especial a la redistribución y modificación del producto. Es más, el usuario que recibe el *software* libre, podrá:

- Usarlo cómo y dónde quiera.
- Redistribuirlo a quien quiera y de la manera que quiera.
- Modificarlo (para mejorarlo, adaptarlo, etc.).
- Redistribuir esas modificaciones.

Estos cuatro puntos se corresponden realmente con las cuatro libertades del *software* libre y que se estudiarán en detalle en la sección 1.3.1 de este mismo capítulo.

Tal y como se puede entender de los cuatro puntos anteriores, será necesario contar con el código fuente.

Y por último, si hay un falso mito en torno al mundo del *software* libre, sobre lo que es y lo que no es, ante todo, debe quedar claro lo siguiente: *software* libre no es igual a *software* gratis. Esta confusión se ha producido a lo largo de los años por el hecho de que habitualmente, los usuarios pueden descargarse el software libre de una manera gratuita. Es más, en inglés el término "free" puede hacer referencia tanto a "libre" como a "gratis", por lo que se genera ambigüedad que si bien en la lengua española no existe, sí se cuenta con esa misma confusión por razones distintas. Este mismo hecho, motivó la aparición de un nuevo término para referirse a lo mismo, tal y como se estudiará más adelante: *Open Source*.

Debido a esas libertades con que contará el *software* libre frente a otros modelos, se tienen una serie de consecuencias como que por ejemplo, el modelo de costes será totalmente distinto al del *software* privativo. También podrá inspeccionarse, estudiarse, modificarse, etc., cosa impensable con el *software* privativo ya que normalmente, es distribuido en forma binaria a través de sus ejecutables. También surgen nuevos canales y métodos de distribución, al mismo tiempo que los modelos de desarrollo son a la vez que revolucionarios, hasta cierto punto sorprendetes; es más, a menudo se incumplen las normas básicas que tradicionalmente se han tenido en cuenta como prácticas ideales en ingeniería del *software*, y no por ello, los resultados finales son de mala calidad. Por último, surge también una fuerte competencia en el campo del mantenimiento y soporte del *software* libre. Es decir, que de esta manera, se combinan dos poderosos mecanismos: la competencia y la cooperación (a menudo, incluso de una manera voluntaria).

Así pues, en este nuevo escenario que se presenta ante el mundo y filosofía del *sofware* libre, aparecen un total de hasta cuatro actores: los usuarios finales del *software* desarrollado, que podrán ser tanto particulares como empresas; el desarrollador o productor del *software*; el integrador de sistemas con el fin último de desarrollar soluciones prácticas

y eficaces; y por el último, el actor encargado de ofrecer mantenimiento, servicio y soporte a ese *software*.

Al mismo tiempo, alrededor del *software* libre aparece todo un listado de términos que pueden causar sobre lo que verdaderamente es o no es. Del ya comentado caso del término "*Open Source Software*" que se estudiará en mayor profundidad a lo largo del capítulo, podemos decir que en general, y salvo excepciones muy puntuales, el *software open source* es también *software* libre. Pero a este término se suma también la idea del dominio público, en el cual se debe tener claro que el autor realmente está cediendo sus derechos a la sociedad. O el término *copyleft*, que hace referencia a que ese *software*, tras una serie de redistribuciones, mantendrá las libertades para quien lo reciba.

1.1.3. Conclusiones

¿Qué conclusiones existen en la actualidad respecto al mundo del *software* libre? Lo cierto es que realmente todavía no se cuenta con la suficiente experiencia como para aventurarse a lanzar un veredicto sobre las tendencias futuras en el *software* libre. Sin embargo, sí existen perspectivas interesantes de cada al futuro, como por ejemplo la buena muestra de casos en los que se ha demostrado la viabilidad económica y técnica del *software* libre. El modelo favorece a los más competitivos, ya que permite estar en la cresta de la ola tecnológica a todos aquellos con verdaderas ganas de innovar y para lo cual es necesario tener un buen (y actualizado) conocimiento del área, facilitando el trabajo y ofreciendo posibilidades incluso a los más pequeños dentro del mercado.

No obstante, todavía quedan problemas por resolver y de los cuales, quizás puedan surgir nuevas oportunidades, así como técnicas con las que experimentar en muy diversas áreas, desde los modelos de desarrollo hasta los modelos de negocio. En cualquier caso, lo que sí parece claro es que se trata de un momento en el que toda la industria puede estar cambiando de paradigma y concepción acerca del sector, como de hecho bien demuestra la cantidad de grandes multinacionales que cada vez con mayor frecuencia e implicación, se acercan al *software* libre.

1.2. Historia del *Software Libre*

1.2.1. Primeros Años

Durante los primeros años de la computación (décadas de 1950 y 1960), la situación del *software* dentro del mundo de la computación era muy similar a la filosofía que hoy conocemos bajo el término "*software libre*". En aquellos años, el *software* (incluyendo su código fuente) estaba incluído de manera inherente a la adquisición del *hardware*, hasta el punto que el *software* no era ni visto ni considerado como una pieza independiente del *hardware*, es decir, que el *software* era una especie de acompañante. Debido a esto, los profesionales de la computación podían mejorar (y así lo hacían) ese *software*, compartiéndolo con otros grupos. Es más, siempre que se pagase el contrato, se tenía acceso al catálogo de *software* del fabricante. Sin embargo, esta situación cambió radicalmente el 30 de junio de 1969, cuando *IBM* (principal fabricante y con mucha diferencia sobre

sus competidores) anunció que a partir de 1970, dejaría de incluir el *software* con sus computadores y comenzó a venderlo de manera independiente. Fue entonces cuando se empezó a ver el *software* como un producto con valor intrínseco, y con ello, se limitaron y restringieron sobremanera las posibilidades que sus usuarios tenían de estudiar el código, mejorarlo, compartirlo, etc.

Con esta situación, a mitad de la década de 1970 ya era completamente habitual encontrarse con *software* privativo, lo que supuso un fuerte cambio cultura entre los profesionales del sector. Pero pese a que la tendencia era hacia la exploración del nuevo modelo ofrecido por el *software* privativo, existieron durante estos años iniciativas importantes que encajan a la perfección con lo que hoy conocemos como *software* libre mediante las cuales se perseguía una herramienta determinada, amparada ya fuera por motivos éticos (costumbre en la comunidad matemática) o prácticos (difusión científica). Algunos ejemplos a destacar son *SPICE* para la simulación de circuitos integrados, *TeX* como sistema de tipografía electrónica y el complejo caso de *Unix*, sistema operativo portable originalmente creado con Ken Thompson y Dennis Ritchie en los *Bell Labs* de *AT&T*.

Pero hasta el momento todo fueran iniciativas prácticamente individuales. No fue hasta comienzo de la década de 1980 cuando comenzaron a aparecer proyectos organizados y que de una manera consciente, buscaban sistemas compuestos únicamente por *software* libre. Fue precisamente durante estos años cuando se asentaron las bases y fundamentos filosóficos y legales (a través de la licencia *GPL*) del movimiento del *software* libre. Richard Stallman al frente, como padre ideológico y cabeza visible del movimiento, dejó en 1984 su puesto de trabajo en el *MIT* para comenzar a trabajar en el proyecto *GNU*, con la idea de crear un sistema *software* de propósito general y completamente libre. De hecho, aportó alguna infraestructura básica para continuar trabajando en el sistema, como fueron el editor de textos *Emacs* y el importante compilador de *C* llamado *GCC*.

Además, Richard Stallman también se encargó de escribir la licencia *GPL* con la que se garantizaba que cualquier usuario, tras un número indefinido de redistribuciones, siguiese contando con las mismas libertades para modificar el *software*, redistribuirlo, etc. Y del mismo modo, fundó la *Free Software Foundation* con tal de conseguir fondos destinados al desarrollo y protección del *software* libre, asentándose las bases del pensamiento y la ética en el *The GNU Manifesto*.

Por otro lado, alrededor de *Unix* se creó una comunidad de desarrolladores que pronto tuvo como centro al *CSRG* de la *Universidad de California* en Berkeley, hasta el punto de llegar a convertirse en una de las dos fuentes principales de *Unix*, junto con la oficial, *AT&T*. De hecho, tal fue el éxito del *software* desarrollado en el *CSRG* que tras liberaciones de su código del núcleo y todas las utilidades de un sistema *Unix* completo, se daría origen primero al sistema *386BSD* (Bill Jolizt escribío el código fuente del núcleo para que funcionase sobre *i386*), y a partir de ahí nacería toda la familia **BSD* (*NetBSD*, *FreeBSD*, *OpenBSD*), siempre distribuida bajo licencias tipo *BSD* y que en muchas ocasiones fue y sigue siendo utilizado por software privativo: *SunOS*, *Ultrix*, etc.

Un hito importante en la historia temprana del *software* libre lo supuso *Internet*, desde su nacimiento a comienzos de la década de 1970 y con el que estableció una estrecha relación. Fue importante la distribución de una implementación libre de la pila *TCP/IP* llevada a cabo por *BSD Unix* y fue tomada como la de referencia. La red pronto se convirtió

en una herramienta fundamental, al permitir la cooperación y compartir información y *software* a través de las mismas y con independencia de la distancia existente entre los interesados. A todo esto pronto se añadieron nuevas herramientos que ayudaron aún más en la metodología de trabajo, como *News*, *FTP*, correo electrónico, etc. Y con todo ello, comenzaron a formarse comunidades alrededor de proyectos de *software* libre tal y como hoy las conocemos.

1.2.2. Final de los 1980 y década de 1990

Hacia finales de los años 1980 ya se contaba con una gran cantidad de aplicaciones y herramientas necesarias en un sistema *software*. De hecho, muchas de ellas se convirtieron rápidamente, dada su calidad, en las mejores de su campo, como era el caso del compilador *GCC* o ciertas utilidades *Unix*, además del interesante sistema *X Window*.

Durante estos años, nace el interesante caso de la ya comentada familia **BSD*, con los sistemas *386BSD*, *NetBSD*, *FreeBSD* y *NetBSD*, nacidos en el *CSRG* de la *Universidad de California* en Berkeley. Se trata de sistemas formados completamente por *software* libre, tras conseguir el último componente que faltaba para ser así. Concretamente se trataba del *kernel* del sistema, que Bill Jolitz desarrolló para que funcionase sobre *i386* y naciendo así el sistema *386BSD* que daría origen al resto de la familia. Otra característica común a todos ellos es que se distribuyen bajo una licencia *BSD* de tipo permisiva, por lo que puede ser utilizada para distribuición como *software* propietario.

Por su parte, al proyecto *GNU* le seguí faltando también la misma pieza para conseguir el soñado sistema completamente libre: el *kernel*. De hecho dentro del proyecto *GNU* ya se contaba con un proyecto para solventar dicho espacio, conocido como el proyecto *Hurd*. Pero la verdadera revolusión llegó cuando en 1991, un estudiante finlandés de 21 años llamado Linus Torvalds, anunció que estaba trabajando en un nuevo kernel inspirado en el sistema *Minix* desarrollado por Andrew S. Tanenbaum y que gozaba de gran popularidad, siendo principalmente utilizado en entornos académicos con fines educativos. Desde ese mismo momento, centenares de desarrolladores se vuelcan en el trabajo sobre el nuevo kernel llamado *Linux*, e integrándolo junto con el resto de herramientas del proyecto *GNU*, así como creando otras nuevas. Además, resulta de especial importancia que este *kernel* sea distribuido bajo una licencia *GPL* (ahorrándose problemas que tuvieron los sistemas distribuidos bajo licencia *BSD*), por lo que las redistribuciones del mismo deben hacerse con el código fuente. Queda configurado por tanto el sistema operativo completamente libre: *GNU/Linux*. Su popularidad queda confirmada con la aparición de múltiples distribuciones diferentes pese a contar con un mismo *kernel*, como son los famosos y pioneros casos de *Slackware*, *Debian*, *RedHat*, *Suse*, y así sucediéndose unas distribuciones a otras hasta llegar a nuestros días con la conocidísima *Ubuntu*.

Hacia finales de la década de 1990 el *software* libre ya ofrece sistemas completos tanto con *GNU/Linux* como con **BSD*. Sin embargo, sigue habiendo ciertos déficits como por ejemplo los sistemas de ventanas para interfaces gráficas en un tiempo en el que *Windows 95* arrasa en el mercado precisamente gracias a este factor, que hace que pueda llegar de una manera masiva a un público general gracias a su sencillo uso. Pero muy pronto llegarían iniciativas destinadas precisamente a solventar ese hueco dentro de los escritorios con los proyectos *KDE* y *GNOME*. Al mismo tiempo, *GNU/Linux* se torna omnipresente

en los entornos universitarios y comienza a llegar a las propias casas de los estudiantes. Por otro lado, el *software* libre se sigue confirmando como la mejor opción en muchos ámbitos y sectores, como por ejemplo en el segmento de los servidores *HTTP*, que siempre han liderado el mercado gracias a su calidad, como ocurre en el conocido caso de *Apache*. Pero esto mismo aplica a muchos otros ámbitos, como por ejemplo en la propia estructura de *Internet*, o en el mundo de los compiladores.

Otro punto de especial importancia comienza a darse en el ámbito económico y empresarial, al mismo tiempo que en algunos casos también se capta la atención de la prensa. El caso de *RedHat* se convierte en una buena muestra de que se puede crear negocio alrededor del *software* libre, en su caso, comenzando con la venta de *CDs* que incluían instalaciones de *GNU/Linux* y diversificando el modelo con el tiempo. Las grandes empresas comienzan a ser conscientes de la importancia que el *software* libre está cobrando, llegando en muchos casos a definir una política respecto al mismo, como ocurre en los casos de *IBM*, *Apple* o *Corel*.

1.2.3. Década de 2000 y actualidad

Durante el comienzo de la década de los años 2000, se sigue produciendo un gran aumento del número de desarrolladores y de la cantidad de *software* libre disponible, que comienza a estar listo para el escritorio gracias a los proyectos *GNOME*, *KDE*, y a la continua aparición de aplicaciones para usuarios medios, como por ejemplo *suites* ofimáticas (*OpenOffice*) y navegadores *web* (*Mozilla Firefox*, tras toda la evolución desde *Netscape*), al mismo tiempo que las distribuciones se perfeccionan y comienzan a ofrecer una gran facilidad de instalación, hecho llevado a su extremo con *Ubuntu*, aparecida en 2004, pensada y dirigida especialmente para usuarios medios sin conocimientos técnicos.

Del mismo modo, definitivamente las grandes empresas tienen en cuenta el movimiento del *software* libre. En unos casos lo incorporan a sus estrategias de mercado, como ocurre en los casos de *IBM*, *Sun* o *HP*, y en otros, muestran su rechazo al mismo produciéndose cierto enfrentamiento y criticando particularmente al software libre distribuido bajo licencia *GPL*. Además, en medio de las dificultades financiera debidas a la crisis de las *puntocom*, el *software* libre comienza a penetrar también en las administraciones públicas, hasta el punto de que la distribución *gnuLinEx* adoptada en 2002 por la *Junta de Extremadura* se convierte en portada del *Washington Post*, siendo una de las primeras ocasiones en las que el *software* libre aparece en portada de un periódico importante.

Es durante estos años cuando se comienza a estudiar el fenómeno del *software* libre con tal de comprender el movimiento y su funcionamiento un poco mejor. También se hace palpable el efecto de la deslocalización, por el cual, países considerados como periféricos aumentan su contribución y de manera muy interesante al *software* libre. Pero al mismo tiempo, se produce una gran actividad en el ámbito legal. En este caso cuenta con una especial relevancia el caso *SCO*, al imponer a principios de 2003 una demanda a *IBM* por supuesta infracción de propiedad intelectual. El entorno legal empieza a cambiar y sólo el tiempo dictará si será en forma de impedimento y barrera para el *software* libre.

Con el paso del tiempo, el *software* libre se convierte cada vez de una forma más acelerada en una opción para la estrategia de mercado de empresas tan natural como

lo pueda ser cualquier otra. El ejemplo más claro de esto lo protagoniza de una manera evidente *Google*, empleando el *software* libre como estrategia para entrar en un determinado mercado y llegar a dominarlo (casos de *Android*, *Symbian*, *Meego*, etc.). De este modo, se producen dos hitos fundamentales y de gran importancia para la situación actual. Por un lado, el caso de *Ubuntu*, distribución destinada a usuarios medios y que comienza a ser empleada de una manera casi sin precedentes en entornos de escritorio, y el caso de *Android*, haciendo que el *software* libre dé el salto a nuevas áreas y tecnologías, como es en este caso el mercado de la telefonía móvil y los *smartphones*, cosechando igualmente un gran éxito y penetración entre la población.

1.2.4. Futuro

Pese a la dificultad de predecir lo que ocurrirá en un futuro, sí parece claro que el *software* libre deberá hacer frente a una serie de problemas a los que de hecho se lleva enfrentando desde hace años:

- Técnica *FUD* (fear, uncertainity, doubt), consistente en crear miedo, servirse del desconocimiento o sembrar dudas en torno a este modelo. Sin embargo, a medida que el *software* libre evoluciona, se muestra cada vez más inmune a estas técnicas.
- Disolución, consistente en poner a prueba los límites del *software* libre, hasta el punto de ofrecer modelos que presentan características muy similares a las del mismo, con la consecuente creación de confusión, estudio de si realmente se trata de *software* libre con las consecuencias de ello, etc.
- Desconocimiento, ya que no es extraño, y especialmente en los últimos tiempos, encontrar personas que llegan al mundo del *software* libre por moda, o engañados por la falsa idea de asociar *software* libre con *software* gratis. El principal problema aquí es la insatisfacción que se puede producir al desconocer las ventajas que ofrece el modelo y la consecuente explotación de las mismas.
- Impedimentos legales, que parecen ser los mayores problemas de cara al futuro, con las nuevas legislaciones sobre derechos de autor, patentes en el mundo del *software* y que en definitiva, suponen barreras para la entrada del *software* libre en determinados nichos de mercado.

1.3. La Definición de *Open Source*

1.3.1. Motivación

Existe un motivo fundamental para definir aquello de lo que estamos hablando: evitar confusiones. Por ello, en esta sección estableceremos a qué nos estamos refiriendo al hacer mención de aspectos como *Free Software*, *Open Source* o *Software Libre*.

Para ello, debemos recurrir en primera instancia a Richard Stallman y la *Free Software Definition*[1], cuya lectura detenida queda totalmente recomendada. En esta definición, se

[1] http://www.gnu.org/philosophy/free-sw.html

establecen las cuatro libertades que debe ofrecer un *software* para que pueda ser considerado como libre, y que se detallan a continuación:

0. La libertad de ejecutar el programa para cualquier propósito. De lo cual se desprende que no se debería prohibir el uso del programa, por ejemplo, con fines militares, si bien es cierto que esto no ampara los posibles usos para fines ilegales.

1. La libertad para estudiar cómo funciona el programa y modificarlo como se desee. De esto se desprende lo que se podría considerar como un corolario: es necesario tener acceso al código fuente.

2. La libertad para redistribuir copias ya que con ello, puedes ayudar a tu vecino. La manera de redactar esta libertad en concreto, deja entrever la filosofía de vida y pensamiento de Richard Stallman.

3. La libertad para redistribuir copias del *software* una vez modificado. Se puede entender como una combinación de las dos libertades anteriores.

Como análisis de estas libertades, se puede afirmar que Richard Stallman en el momento de su definición, estaba poniendo el énfasis en la idea de la libertad (en el sentido de la libertad de expresión). De hecho, en ningún momento se recogen menciones al ámbito comercial y/o económico, ya que para tener libertad de expresión no es necesario pagar (obsérvese la desambiguación del término inglés *"free"*, indicándose que debe entenderse en el sentido de *"free speech"* y nunca en el sentido de *"free beer"*, por lo que igualmente se hace énfasis de una manera indirecta en que por el hecho de ser libre, el *software* no tiene por qué ser gratuito).

En cualquier caso, la *Free Software Definition* es tan sólo una definición que nos ayuda a identificar y concretar a qué nos referimos al emplear el término *Free Software*. Pero a la hora de materializar este concepto en la práctica, ¿cómo ha de hacerse? Debido a la propiedad intelectual y sus mecanismos, si no se da permiso explícito, esa obra pertenece a su creador y nadie más puede hacer uso de ella.

De aquí nace la necesidad de contar con el concepto de licencia, mecanismo empleado para indicar al usuario qué puede hacer y qué no con ese *software*, y que se estudiará en detalle en el capítulo 2 de este documento. De este modo, será necesario leer la licencia de ese *software* y en caso de cumplirse con las cuatro libertades indicadas anteriormente, nos encontraremos ante un *software* libre. De lo contrario (bastará con que no se cumpla tan sólo una de las cuatro libertades) estaremos ante un *software* privativo o *software* propietario (estos dos últimos términos se emplean como sinónimos, pese a que lo correcto es el término privativo, ya que por el hecho de hacer que un *software* sea libre, no se pierde la autoría desde el punto de vista legal, se continúa siendo el propietario de cara a la ley).

En cualquier caso, resulta indudable que la lectura y análisis de una licencia *software* es un trabajo tedioso. Por ello, es preferible contar con un listado de licencias asimiladas y reconocidas como de *software* libre. Además, este listado de licencias deberá ser emitido por una entidad de reconocido prestigio en la materia.

Retomando la definición ofrecida por la *Free Software Foundation*, debe notarse que se trata de un lenguaje abstracto y de altísimo nivel, mientras que una licencia emplea

un lenguaje muy concreto de lo que se puede y lo que no se puede hacer. Este fue el motivo por el que nacieron las *Debian Free Software Guidelines*[2] (lectura totalmente recomendada): dado que la definición original es de muy alto nivel, se deben establecer reglas más detalladas, de tal modo que se pase de hablar de aspectos como la libertad de expresión, a cosas mucho más concretas.

Este trabajo llevado a cabo por *Debian* tuvo gran éxito y buena acogida, y por ello fue tomado como base para otras definiciones, como es el caso de la *Open Source Definition*. Pero, ¿por qué es necesaria una nueva definición para una misma cosa? La razón de esto se encuentra en que pese a todas las matizaciones e indicaciones, *Free Software* puede referirse en inglés tanto a *"libre"* como a *"gratis"*. Por ello comenzó también a emplearse el término *Open Source*, dándose origen a la *Open Source Definition*, que vamos a estudiar en detalle a continuación.

1.3.2. *The Open Source Definition (Annotated)*

Tomando como base las *Debian Free Software Guidelines*, y con tal de desambiguar el término *"free"* en inglés (contando con la polisemia *"gratis"* y *"libre"*), nació la *Open Source Initiative* (en un primer momento, intentando incluso la creación de *Open source* como marca comercial, cosa que no se consiguió debido a que se trataba de un término demasiado genérico) con su *Open Source Definition*[3] que será analizada durante esta sección, y que comienza de la siguiente manera:

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

Aquí se aclara que para que un *software* se pueda considerar como *Open source*, no se debe cumplir únicamente con que se tenga acceso a su código fuente (hecho que por otro lado, también genera ciertas confusiones en la actualidad pese a la clara matización que se ofrece).

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

Rationale: By constraining the license to require free redistribution, we eliminate the temptation to throw away many long-term gains in order to make a few short-term sales dollars. If we didn't do this, there would be lots of pressure for cooperators to defect.

Como se puede observar en este primer punto, y al contrario de lo que ocurría con la *Free Software Definition*, en esta definición se comienza a hilar muy fino y tratando

[2] http:www.debian.orgsocial_contract#guidelines
[3] http://www.opensource.org/docs/definition.php

temas muy concretos. Este punto en concreto trata sobre el hecho de que la licencia no debe restringir la distribución de un *software* como parte de una distribución de software agregado o global, conteniendo programas de diferentes fuentes, al mismo tiempo que el creador original tenga el derecho a exigir *royalties*, como por ejemplo una determinada cantidad económica por cada descarga, etc.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

Rationale: We require access to un-obfuscated source code because you can't evolve programs without modifying them. Since our purpose is to make evolution easy, we require that modification be made easy.

En este segundo punto se indica que será necesario incluir el código fuente en la distribución del *software*, y en caso de que no sea así, se debe informar claramente de las diferentes maneras existentes para obtenerlo, nunca por más de un precio razonable en concepto del coste de reproducción, o a través de *Internet* sin coste adicional. Otro aspecto a destacar en este punto sobre referencias a aspectos concretos, es la referencia a que el código fuente no se debe ofuscar de manera deliberada (cambiar los nombres de las variables por otros nombres aleatorios, eliminar espacios y tabuladores alineando el código fuente en una única línea de código, etc.) con tal de que la legibilidad del código se mantenga para facilitar el estudio del mismo, etc.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

Rationale: The mere ability to read source isn't enough to support independent peer review and rapid evolutionary selection. For rapid evolution to happen, people need to be able to experiment with and redistribute modifications.

La licencia debe permitir modificaciones y obras derivadas, y debe permitir su redistribución bajo las mismas condiciones. Aquí se encuentra un matiz importante, ya que recurriendo a aspectos de propiedad intelectual, se podría "obligar a", mientras que se prefiere utilizar "te permito hacerlo". De hecho, cabe comentar la existencia de licencias de *Software Libre* que no obligan a mantener la misma licencia, de modo que se pueden generar obras derivadas privativas, tal y como ocurre en el caso de las licencias tipo *Apache*.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

Rationale: Encouraging lots of improvement is a good thing, but users have a right to know who is responsible for the software they are using. Authors and maintainers have reciprocal right to know what they're being asked to support and protect their reputations. Accordingly, an open-source license must guarantee that source be readily available, but may require that it be distributed as pristine base sources plus patches. In this way, "unofficial" changes can be made available but readily distinguished from the base source.

Se permite la distribución y modificación del *software*, pero de tal modo que la obra original quede íntegra, mientras que las modificaciones se realicen a modo de parches, pudiéndose realizar obras derivadas, pero no como la obra original (caso de *Apache*, y sus modificaciones como pueda ser el ejemplo de *Cherokee*). Este aspecto hace referencia a la práctica es habitual consistente en el envío de parches al autor original para que los incluya con tal de no tener que volver a aplicarlos con nada nueva *release* de dicho *software*.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

Rationale: In order to get the maximum benefit from the process, the maximum diversity of persons and groups should be equally eligible to contribute to open sources. Therefore we forbid any open-source license from locking anybody out of the process.
Some countries, including the United States, have export restrictions for certain types of software. An OSD-conformant license may warn licensees of applicable restrictions and remind them that they are obliged to obey the law; however, it may not incorporate such restrictions itself.

Este punto (así como el punto siguiente) hace referencia a la libertad 0 indicada por la *Free Software Definition*, de tal modo que no se debe discriminar a ninguna persona bajo ningún criterio (sexo, religión, raza, nacionalidad, ideología política, etc.). Pero en cualquier caso, el hecho de que el *software* sea libre, no quiere decir que tus actos con el mismo sean legales, como pueda ser la exportación de armas, etc.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

Rationale: The major intention of this clause is to prohibit license traps that prevent open source from being used commercially. We want commercial users to join our community, not feel excluded from it.

Haciendo referencia también a la libertad 0, la licencia no debe restringir el uso del programa en un determinado campo de actividad.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

Rationale: This clause is intended to forbid closing up software by indirect means such as requiring a non-disclosure agreement.

En este punto, lo importante es que no se pueden incluir condiciones adicionales que puedan restringir las libertades del *software*.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

Rationale: This clause forecloses yet another class of license traps.

En este punto se indica que no se puede permitir la distribución como *software libre* para un determinado producto, pero no para otros (por ejemplo, no se puede indicar que un *software* se puede distribuir como libre bajo una distribución *Debian*, pero no bajo una distribución *Red Hat*). Es decir, no se pueden producir acuerdos específicos, sino que las reglas deben ser las mismas para todos.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

Rationale: Distributors of open-source software have the right to make their own choices about their own software.
Yes, the GPL v2 and v3 are conformant with this requirement. Software linked with GPLed libraries only inherits the GPL if it forms a single work, not any software with which they are merely distributed.

Para entender esta regla en su integridad, será necesario el estudio de la licencia *GPL* en el capítulo 2.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

Rationale: This provision is aimed specifically at licenses which require an explicit gesture of assent in order to establish a contract between licensor and licensee. Provisions mandating so-called "click-wrap" may conflict with important methods of software distribution such as FTP download, CD-ROM anthologies, and web mirroring; such provisions may also hinder code re-use. Conformant licenses must allow for the possibility that (a) redistribution of the software will take place over non-Web channels that do not support click-wrapping of the download, and that (b) the covered code (or re-used portions of covered code) may run in a non-GUI environment that cannot support popup dialogues.

Se trata de otro punto peculiar dentro de la definición, según el cual, la licencia no debe presuponer ninguna tecnología, cosa que se entiende fácilmente en los dispositivos empotrados (lavadora, TDT, etc.), ya que normalmente no se dispondrá de una interfaz adecuada (*display*, botón de *OK* o aceptar, etc.) como para visualizar los términos de la licencia del *software* para aceptarlos o rechazarlos.

1.3.3. Comparación de Diferentes Definiciones

Llegados a este punto, debemos realizarnos la siguiente pregunta: ¿en qué se diferencian las diversas definiciones de *software libre*? Durante esta sección, intentaremos responder a la pregunta.

Las diferencias no van más allá de un mero aspecto filosófico. Así, *Free Software* es el término empleado por aquellos que tratan sobre la parte más política y cercana a los aspectos de libertad de expresión, mientras que el término *Open Source* se ha popularizado en los círculos de ámbito comercial, quedando quizás más próximo a los aspectos puramente pragmáticos del *software libre*.

Sin embargo, *Free Software* y *Open Source* no son más que maneras diferentes de referirse a una misma cosa, hasta el punto de que prácticamente las podemos usar como sinónimos. De hecho, personas con diferentes maneras de pensar, *a priori*, respecto a un mismo tema, materializan sus ideas en la práctica de una misma forma: las licencias de *software libre*. La mejor prueba para demostrar la equivalencia entre los términos es que las listas de licencias aceptadas como de *software libre* ofrecidas por unas y otras instituciones son prácticamente exactas salvo en casos muy puntuales. Precisamente esas excepciones, se deben a las diferentes maneras posibles de interpretar una misma cosa, pero siendo siempre coincidentes en la esencia.

En definitiva y a modo de conclusión, debemos tener en cuenta que *Free Software* y *Open Source* no son dos cosas diferentes. De la misma manera que tampoco lo son los términos *Libre Software* (nacido igualmente para evitar la confusión a la que puede llevar el término *"free"* del inglés, de modo que se adoptó la palabra del español) y el más reciente *FLOSS* (*Free, Libre Open Source Software*) nacido como una manera de unificar criterios con tal de dar origen a un término único como forma de referirse y denotar al mundo del *software libre*.

Capítulo 2

Aspectos Legales

En este capítulo se verán los elementos principales involucrados en los aspectos legales del software libre. El más importante de ellos es la licencia, que supone el contrato establecido entre el autor y los usuarios y que define qué libertades tienen estos últimos sobre la obra en cuestión. Antes de entrar en detalle con las licencias se dará un barrido sobre la propiedad intelectual, que engloba conceptos como *Copyright* y patente, entre otros.

2.1. La propiedad intelectual

La propiedad intelectual engloba los derechos de autor, las patentes, el secreto industrial y las marcas. Es compleja, principalmente por la dificultad de marcar sus límites. Prueba de ello es el caso del puente Zubi Zuri bilbaíno, en el que Santiago Calatrava, diseñador del puente, denunciaba las modificaciones que el ayuntamiento de la ciudad efectuó sobre su obra. En una primera instancia, la reclamación del pago de una indemnización de 250.000 euros fue denegada. Tras la presentación de un recurso, el arquitecto recibió 30.000 euros. Esta sentencia deja la sensación de que el bien público se antepone sobre la propiedad intelectual. Hay otros casos interesantes como la demanda que el artista Nach interpuso sobre el Ministerio de Sanidad, motivado por un anuncio que fomentaba el uso del preservativo. Tanto el tema "Efectos Visuales" del cantante como la canción del anuncio compartían estilo músical y versos en los que se utilizaba una única vocal. La petición fue desestimada. El omnipresente Google también ha sufrido demandas de este tipo. Su programa de recopilación de noticias GoogleÑews hacía que los periódicos autores de las noticias perdieran ingresos al disminuir los interesados en publicitarse en sus páginas oficiales. Como se puede observar, la propiedad intelectual afecta a muchos sectores. En las siguientes secciones veremos cómo nace el concepto, las diversas interpretaciones geográficas y las prácticas habituales para salvaguardarla.

2.1.1. Orígenes de la propiedad intelectual

La propiedad intelectual nace junto con uno de los mayores inventos de la historia: la imprenta. Hasta ese momento, la copia manual era la única forma de reproducción de obras

pósible. Los autores ni siquiera se habían planteado el problema que vendría años después en Inglaterra.

La imprenta ofrecía la posibilidad de reproducir libros con gran rapidez, lo que desembocó en nuevos modelos de negocio. La venta de libros y periódicos a gran escala se convertía así en algo factible. Los dueños de las imprentas empezaron a rentabilizar sus máquinas. Los autores, conscientes del nuevo mercado, exigían su parte de beneficios. Para ello tenían que llegar a acuerdos con los impresores. Pero, *¿qué pasaba si otra imprenta se hacía con una copia y la reproducía sin llegar a un acuerdo con el autor?* Para solventar ese problema, los ingleses realizaron "pactos de caballeros" mediante los que no se publicaría ninguna obra sin el consentimiento del escritor. Pronto llegarían los galeses para saltárselos.

Los empresarios ingleses empezaron a manifestar su animadversión contra sus competidores galeses. Ellos no pagaban a los autores, por lo que su margen de beneficios era mayor. Para hacer frente al problema, acudieron a la Reina Ana de Gran Bretaña, máxima autoridad de la época, y expresaron sus quejas. La Reina encontró una solución para los impresores ingleses y sobre todo para sí misma: impuso un permiso de copia que sólo podía ser otorgado por la Corona. Con esta decisión adquiere un control total sobre los libros que se imprimían, que podrían contener ideas subversivas contra el reino. Nace así la censura en las imprentas.

Más tarde, surge en Francia una nueva rama de la propiedad intelectual. Consiste en la protección del autor por encima del resto de factores. Supone que el artista es tocado por Dios, lo que le permite crear su obra. Esta rama ha tenido una gran influencia en nuestra legislación actual[1], en la que se separa *propiedad intelectual* de *propiedad industrial*. Una vez introducido el origen de las prácticas, pasamos a ver uno de los elementos que más perjudica al Software Libre, las patentes.

2.1.2. Las patentes

Una patente otorga el monopolio del invento a su creador durante un periodo que va desde 17 a 25 años, según la región. El inventor ha dedicado un tiempo al trabajo y recibe esta recompensa. A cambio, el invento debe ser públicado. Las terceras partes que quieran trabajar sobre él deben llegar a un acuerdo con el titular de la patente.

No se puede patentar una idea. La institución registradora de patentes se encarga de acotar los límites del invento. *La patente de una silla, ¿debería permitirme explotar también sillones y sofas?* Sin embargo, hay organizaciones como la *Organización Mundial de la Propiedad Intelectual* (OMPI) que están ejerciendo presión para que se relajen las restricciones de registro de una patente. No es un procedimiento al alcance de cualquiera por lo que unas pocas empresas suelen ser dueñas de la inmensa mayoría de patentes. Por otro lado, se ha detectado una práctica denominada *patent trolling* que engloba a las entidades que se encargan de patentar el mayor número posible de inventos, no para explotarlos y lucrarse con ellos, sino para obtener beneficios mediante los acuerdos con interesados en el invento.

En el software libre las patentes suponen un problema muy grande. El acceso al código fuente facilita la detección de infracciones sobre patentes. Los autores de software privativo

[1] En palabras de Grex: "¡nuestros artistas están afrancesados!"

tienen más ventajas en este sentido. En Europa, a diferencia de Estados Unidos, todavía no es posible patentar software, aunque podría llegar ese momento.

2.1.3. El secreto comercial

Las empresas tienen el derecho a no hacer público su trabajo. Un ejemplo muy claro de este tipo de práctica es la ocultación de la fórmula de la *Coca-Cola*. A la compañía no le interesa una patente, ya que saben que pueden rentabilizar su producto mucho más allá de la vigencia de ésta. Otro ejemplo es la ocultación de la ingeniería utilizada entre las diferentes escuderías de la *Fórmula 1*, donde el prestigio por ganar el campeonato tiene un peso muy grande. En ambos casos se pretende esconder los secretos de la empresa para dificultar el trabajo a los competidores.

El secreto comercial resulta más limpio que una patente, en términos de mercado. La razón es que permite la coexistencia con terceras partes que se dediquen a imitar el producto original. No hay más que ver las infinitas variedades de refrescos de cola que se pueden adquirir, habiéndose convertido *Pepsi* en el adversario más fuerte. En la imitación tiene una gran importancia la ingeniería inversa, que permite deducir el proceso de manufacturación a partir del producto final. Los secretos mejor guardados se encuentran en las grandes compañías, que como viene siendo habitual, ejercen mucha presión sobre el Estado cuando se ven amenazados. Por eso, existen regiones en las que la ingeniería inversa está prohibida.

En el software, el proceso de creación de un producto es encarnado por el código fuente. Lo que permite el secreto comercial es la distribución del binario. Por lo tanto, está práctica carece de sentido en el software libre, ya que aplicarla iría en contra de las libertades básicas al no disponerse del fuente.

2.1.4. Las marcas

Una marca es el nombre —y opcionalmente el logotipo— que utiliza un negocio para promocionarse. Registrar una marca requiere cierto desembolso económico, por lo que generalmente no suele realizarse. Esto explica en cierta manera por qué es posible que existan cientos de bares “Pepe”, pero dificilmente se encontrará una tienda deportiva que utilice el nombre “Nike”.

En el software libre, son sólo los grandes productos y compañías los que las utilizan. Entre ellos encontramos a Debian, GNOME o GNU. En la historia del software libre se han encontrado problemas por el no registro de una marca. El caso más representativo es el de Linux, en el que varias personas registraron esa marca y reclamaron royalties a las distintas distribuciones existentes. También es un problema de marcas el que tuvieron Firefox y Debian. Esta distribución se negó a incluir el navegador por la falta de libertad sobre los logos oficiales y el nombre de Firefox. La solución fue incluir el navegador con el nombre y el logotipo renombrados, dando como resultado el proyecto IceWeasel.

2.1.5. Derechos de autor

Los derechos de autor surgen para recompensar a los autores de libros o de arte. Representan lo que también es conocido como *Copyright* y tratan de proteger la expresión de un contenido, no el contenido en sí mismo. Por dar un ejemplo actual, todos los días vemos en los periódicos artículos que tratan sobre los mismos sucesos, pero contados de forma diferente. Si se protegiese el contenido, la noticia sólo podría ser publicada en un periódico.

En España, la encargada de estos derechos de autor es la denominada Ley de Propiedad Intelectual. Esta ley hace una subdivisión en Derechos Morales y Derechos Patrimoniales. Los primeros hacen posible la autoría de una obra y que se respete su integridad. Son vitalicios o indefinidos. Los segundos son los que permiten al autor la explotación económica de su trabajo. Tienen un periodo de vigencia de 70 años[2]. Actualmente, la forma de obtener los derechos de autor es automática. En cuanto se crea una obra, entra en vigor. El autor de un garabato en una pizarra dispone de todos los derechos sobre él. Hace años no se aplicaban por defecto. Es curioso el caso de la mítica película de título *La noche de los muertos vivientes* en la que olvidaron incluir la (C) de Copyright, por lo que directamente pasó a formar parte del dominio público.

2.1.6. ¿Qué pasa con el Software?

Hasta el momento hemos visto diversos mecanismos para tratar la propiedad intelectual. *¿En cual de ellos encaja el Software?* En un principio hubo mucha polémica por decidir si se debían aplicar patentes o derechos de autor, finalmente se impuso el segundo de ellos. No resultaría mantenible solicitar una patente por cada commit realizado. Además existían algunas similitudes con las novelas y dicha legislación había funcionado bien hasta el momento. No tardarían en aparecer incongruencias y situaciones especiales. Y es que el software tiene unas características que le hacen muy diferente del resto de obras existentes hasta la fecha.

2.2. Licencias

Si navegando por *la Red* descubrimos un software sin licencia, *¿qué podríamos hacer con él?* Acabamos de ver en 2.1.6 que lo que aplicamos al software son los derechos de autor, es decir, la obra automáticamente pasa por defecto a restringir todos los derechos de los usuarios. Por tanto, si no encontramos la licencia en dicho programa, debemos asumir que no podemos hacer absolutamente nada con él. Bien es cierto que sería difícil que un juez declarara culpable a una persona por usar ese software. Sin embargo, la modificación y redistribución ya sí que sería un terreno muy pantanoso.

Entonces, *¿qué es una licencia?* Es un contrato mediante el cual el autor se comunica con los usuarios finales. Aunque esto a veces es difícil de entender, la licencia no restringe,

[2]Otro ejemplo de presión sobre la legislación de la propiedad intelectual es la que ejerce Mickey Mouse. Misteriosamente, la ley se prolonga cuando este simpático ratón está a punto de pasar al dominio público.

sino que se encarga de otorgar libertades. Sin ella, comprar un programa no nos daría la capacidad de usarlo tan siquiera, se debe recordar lo que se comentó en el párrafo anterior. La licencia es un contrato muy especial, un contrato que ni firmamos ni aceptamos explícitamente. En ocasiones el instalador pide la confirmación mediante la típica checkbox *I Agree*, pero hay casos en los que el entorno no permite esta posibilidad, como por ejemplo el software embebido en una lavadora.

Las licencias son textos largos y complicados. *¿Qué pasa si no entendemos alguna en concreto?* Siempre que se produce un acuerdo, ambas partes deben ser conscientes de las implicaciones que conlleva. Por esa razón los bancos encuentran muchos problemas con sus clientes, hasta tal punto que algunos acuerdos requieren que el interesado tenga un cierto nivel educacional. En el caso del software la solución es más sencilla. Si no entendemos la licencia, se debe suponer la establecida por defecto, con lo que no se tendría derecho a nada. Otro aspecto habitual es el de encontrar una licencia en otra lengua. Por ejemplo, la FSF sólo considera como válida la GPL en inglés. Las traducciones pueden dar lugar a segundas interpretaciones, a veces incluso provocadas. Ante este caso, siempre hay que quedarse con la interpretación más restrictiva o consultar directamente al autor si fuese posible.

Licenciar software con una licencia libre básica es sencillo. Tan sólo debemos incluir las siguiente líneas al principio de los ficheros fuente del programa:

> Copyright (c) 2010 Foobar Developers. All rights reserved.
> Redistribution and use in source and binary forms, with or without modification, are permitted provided that the redistributions of source code must retain the above copyright notice.

Con la primera línea se está "marcando el terreno". En ella se puede apreciar el año de creación, el nombre del programa y el nombre de los autores. La segunda parte del texto es la que otorga al usuario final la libertad de uso, redistribución, modificación y redistribución de la modificación (principios básicos del software libre). Resulta curioso observar como primero se reservan todos los derechos para posteriormente concederlos. La única restricción que se impone al usuario —en este ejemplo particular— es la de mantener la primera línea tal cual, con el fin de preservar la autoría. En los siguientes apartados se describirán las distintas familias de licencias, nombrando para cada una sus ejemplares más representativos.

2.2.1. Licencias Académicas

Son también conocidas como licencias Minimalistas o de tipo BSD. Son muy adecuadas para investigaciones académicas, generalmente financiadas por el Estado. Pretenden tener un alcance global, por lo que no desean cerrar ninguna puerta. Esto conlleva otorgar todas las libertades al usuario final. La única condición a cambio es la de conservar la autoría de los creadores. Un ejemplo de software en el que se aplica esta licencia es el protocolo TCP/IP. Es un protocolo utilizado por cualquier Sistema Operativo moderno, tanto libre como privado. Las libertades que ofrece esta familia rozan las que podemos encontrar en un software de dominio público.

Licencia BSD Es la más representativa de esta familia. No ofrecen ninguna garantía al usuario final. Éste es una aspecto que siempre ha generado cierta controversia, aunque es bastante comprensible[3]. La permisividad de esta licencia provoca que el software se pueda redistribuir tanto en versión de código fuente como en binario. Para el primer caso, se debe mantener la línea del Copyright y la lista de condiciones en los ficheros fuente. Con el formato binario se debe reproducir de alguna manera el mismo texto que para el caso anterior. Por ejemplo, Windows muestra la nota BSD durante la carga del sistema. Se puede leer la plantilla de esta licencia en `http://www.opensource.org/licenses/bsd-license.php`.

2.2.2. Licencias Permisivas

Las licencias permisivas son muy similares a las licencias académicas, hasta tal punto que podríamos ver a estas últimas como un subconjunto de esta nueva familia.

2.2.3. Licencias Copyleft

Copyleft débil

Copyleft fuerte

[3]Los desarrolladores de software propietario tan sólo garantizan que el soporte es correcto y que el programa se ejecuta.

Capítulo 3

El desarrollador y sus motivaciones

Capítulo 4

Aspectos económicos y de negocio

Capítulo 5

Desarrollo de software libre

Capítulo 6

Análisis de proyectos

6.1. Introducción

Los proyectos de software libre son públicos por naturaleza y podemos tener una serie de datos que podemos encontrar y comparar con otros proyectos.

Como todo análisis empírico tenemos que seguir una metodología:

1. El proyecto que queremos estudiar.

2. Identificar la fuente de datos. Seguir el sistema de gestión de incidencias, recuperar los datos, y una vez que hemos realizado la gestión de incidencias y la recuperación de datos tenemos que realizar la minería de datos, es decir, realizar una técnica descriptiva de un proyecto.

3. Realizar el análisis. Extraección la información que no es evidente, aportándonos más información. Por ejemplo, en que meses del año se realiza más trabajos en el proyecto.

4. Realizar un informe explicativo. Una vez realizado nuestro análisis tenemos que componer un informe que explica toda la información del proyecto.

6.2. Herramientas para el análisis

Para realizar un análisis de proyecto necesitamos un conjunto de herramientas automáticas para evitar errores, para reducir tiempos, y para la replicabilidad. La replicabilidad nos permite construir un proyecto a partir de otro y no empezar de cero ahorrándonos un valioso tiempo. Es deseable que las herramientas que vamos a utilizar sean software libre porque es más fácil

de integrarlas, extenderlas y aplicarla a nuevas funcionalidades. Para realizar cualquier tipo de análisis estadístico tenemos una librería de R. R es un software libre que sirve para el análisis estadístico y gráfico en un entorno de programación. Además nos permite cargar diferentes bibliotecas o paquetes con finalidades específicas de cálculo o gráfico. Por otro lado necesitamos soporte de administración de herramientas estadísticas, esto se traduce a tener un ordenador potente o un servidor potente (gran capacidad de disco duro, ram, cpu potente, etc).

Qué podemos usar o aplicar para realizar un análisis:

1. Vamos a utilizar una base de datos, MySQL/SQLite.

2. Una herramienta que extrae la información de código fuente de los registros y la almacena en una base de datos, CVSAnalY.

3. Y un analizador estadístico, GNU R.

Para la estación de datos vamos a extraer información de repositorios públicos como son CVS, SVN y GIT. Nos proporciona un mecanismo automático que apuntando a una url nos trae la información, la parsea y la guarda en una base de datos o en un fichero para poder trabajar.

Podemos extraer datos para las acciones de desarrollo, lo vamos a tener en una tabla de la base de datos con nombre "scmlog". Tenemos datos registrados de archivos que está en la tabla "file". Otra tabla para los datos de las personas involucradas en el proyecto "people".

6.3. Algunos análisis