TRABAJO FIN DE MASTER EN ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS

Centro Internacional de Postgrado

UNIVERSIDAD DE OVIEDO

FACULTAD DE ECONOMÍA Y EMPRESA

SOFTWARE LIBRE: COSTE, VALOR Y ESTRATEGIA

AUTORA:

"REBECA GARCÍA FUENTES"

TUTOR:

"JESÚS GARCÍA GARCÍA"

Oviedo, Julio 2014

AGRADECIMIENTOS

Dedico este proyecto, a mis queridos padres, María del Carmen y Pedro, mi pareja Iván y mi hermano, Pedro, por todos sus sacrificios y su apoyo en los momentos más difíciles.

Así mismo, también quiero agradecer a Jesús García García por su formación y ayuda en el desarrollo de este trabajo.

Índice General

Índice de Gráficas

Índice de Tablas

1 INTRODUCCIÓN Y ANTECEDENTES

Con este proyecto vengo, además de poner fin a mis estudios de postgrado en Administración y Dirección de Empresas, a demostrar que el uso de software libre puede ser un gran aliado en la reducción de costes, tanto globalmente, como se demostrara en apartados posteriores, donde se estima que el uso del software libre ha supuesto un ahorro para la economía de la Unión Europea de al menos 114.000 millones de euros al año. Así como, su uso en las Administraciones Públicas de nuestro país el cual ha supuesto grandes beneficios. Sin ir más lejos, en nuestra propia provincia el uso de software libre ha reducido en un 35% los gastos anuales. Sin olvidar su potencial estratégico, como fuente de negocio para grandes empresas, como es el caso de Google, a la cual el desarrollo de software libre le ha conferido una posición ventajosa en el mercado con la que está preparada para afrontar la nueva etapa del uso de las tecnologías.

Este proyecto cuenta con diversos apartados, el principal corresponde con el cálculo del valor real que ha supuesto para la economía europea la adopción del software libre, basado en datos recogidos de varias encuestas sobre reutilización de código y combinando con un conjunto de diversas estimaciones macroeconómicas.

Además de este apartado, el proyecto cuenta con una introducción sobre el software libre, abordando sus principales características, así como su evolución en el mundo, sin obviar su uso en las Administraciones Públicas, por ser este uno de los puntos claves para establecerse una posición en el mercado.

Finalmente el proyecto concluye con un caso de estudio, basado en la utilización del software libre como estrategia empresarial, como es el caso de Google, empresa mundialmente reconocida, la cual es creadora de Android, sistema operativo presente en la mayor parte de los dispositivos móviles actualmente utilizados.

2 SOFTWARE LIBRE

Si nos ubicamos en el mundo de la informática, cuando un usuario compra un programa o un conjunto de ellos, denominado paquete, recibe este producto en código binario[1], es decir en lenguaje de programación. Este consiste en un sistema de numeración en el que los números se representan utilizando simplemente las cifras cero y uno, por lo que es inaccesible si se desea introducir cualquier tipo de modificación.

Por lo tanto el usuario tan sólo está comprando una licencia que le permite hacer uso de ese programa. Lo que implica una dependencia permanente del proveedor, dado que cualquier modificación que el usuario quiera plantear, requiere acceder al código fuente o código original. Debemos de tener en cuenta que el código fuente o código original, es un conjunto de líneas de texto, las cuales son las instrucciones que debe de seguir el ordenador, por lo que en el código fuente se presenta por escrito el funcionamiento del software.

Por lo general, la mayor parte de los países incluyen la producción y distribución de software bajo régimen de propiedad intelectual o de patente de invención. Pero tal protección no resulta eficaz para resguardar los derechos sobre un producto que, por su naturaleza de intangible, sería susceptible de libre apropiación y adaptación. Por ello, la retención del programa en código fuente es la única vía por la que el propietario puede proteger su creación. De modo que se entiende que el usuario sólo reciba la versión en código binario, el cual no permite realizar ningún tipo de modificación, así tradicionalmente el desarrollador individual o empresa trata de reservarse en exclusiva la explotación económica del software.

Podemos concluir que a la ahora de desarrollar un software, los desarrolladores crearan el llamado código fuente, el cual está escrito en algún tipo de lenguaje de programación, solamente accesible para el propio programador, que incluso no es directamente ejecutable por el ordenador. Mientras que el software que el usuario adquiere, está

[1] El sistema binario se presentó por primera vez, por un antiguo matemático indio Pingala, en el siglo tercero antes de nuestra era. Pero fue en 1854, el matemático George Boole quien desarrolló un sistema de lógica llamado "Álgebre de Boole" que permitió el desarrollo del sistema binario actual.

escrito en otro lenguaje, llamado lenguaje máquina o código objeto, el cual si puede ser ejecutado por el hardware de su ordenador.

Todo lo comentado anteriormente genera una dependencia absoluta del proveedor, la cual, se ha caracterizado tradicionalmente el mercado del software, dando lugar a situaciones más o menos críticas de acuerdo con las características y la predisposición del proveedor. Veamos que según la filosofía del proyecto GNU, por la cual, el software libre es un movimiento social, mediante el que se considera que todo aquel software que no es libre, es un problema social, y la solución es el software libre. Por lo tanto se considera inmoral la instalación de un software privativo, según su fundador Stallman (2002) cuando un usuario adquiere un software privativo genera un conflicto ético, ya que si este se lo enseña a un amigo y el amigo entusiasmado con el producto, desea una copia del mismo. El usuario, se enfrenta a un problema ético, ya que, o bien es un mal amigo por no dejarle el software, o es un mal cliente si por el contrario le deja el software.

Podemos observar que este tipo de dilema o conflicto es a un nivel personal pero más adelante podremos estudiar los conflictos o vulneraciones que hay detrás de un software privativo.

2.1 ¿QUÉ ES EL SOFTWARE LIBRE?

Frente a la situación descrita anteriormente, conocida como software propietario, cerrado o privativo[2]. Surge el concepto de código abierto (Open Source Software) que brinda la posibilidad que de los usuarios tengan acceso al código fuente y lo modifiquen sin intervención del proveedor. La idea general es que el código fuente no solamente debe estar a disposición de cualquier usuario, sino que no debe de tener coste significativo.

El usuario podrá adaptarlo a sus necesidades específicas y redistribuirlo. Para la redistribución es necesario transmitir tanto el código original como el modificado al resto de los usuarios, bajo las mismas normas.

Esta forma de uso da lugar a que el software de código abierto se desarrolle, en general a través del esfuerzo colectivo, aunque no necesariamente coordinado, de cientos, o incluso miles de programadores, al conjunto de participantes se les conoce como "comunidad", la cual esta formada por empresas no lucrativas, programadores individuales y los mismos usuarios. Todos ellos van generando y desarrollando en el software inicial, nuevas modificaciones, adaptaciones o mejoras, lo cual da lugar a un software de gran calidad.

[2] El software privativo es software cuyo código está oculto y su uso, visualización, redistribución o modificación esta prohibida o requiere de autorización.

En 1984, un programador estadounidense, Richard M. Stallman y el proyecto GNU lideran el movimiento del software libre, creándose más tarde la Fundación de Software Libre (FSF). Stallman considera que el software privativo no siempre respeta las libertades de los usuarios, ya que estos solo consideran un software como una herramienta de trabajo.

Se considera que un software es libre se están identificadas en él, estos cuatro principios básicos de libertad:

Libertad 0. -La libertad de ejecutar el programa, con cualquier propósito.

Libertad 1.-La libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades. El acceso al código fuente es una condición previa para esto.

Libertad 2.-La libertad de redistribuir copias

Libertad 3.-La libertad de mejorar el programa, y publicar sus mejoras al público, de manera que beneficie toda la comunidad. El acceso al código fuente es una condición previa para esto.

Esta novedosa modalidad de desarrollo permite cumplir los requisitos básicos de la elaboración del software: la calidad, la velocidad y la independencia en la revisión. Pero curiosamente el proceso de elaboración no se basa en normas o documentos formales, como prescriben las metodologías tradicionales de diseño y desarrollo de software, sino en usos y costumbres implícitos aprendidos a través de la experiencia. Conforman así comunidades virtuales, donde sus miembros operan en forma cooperativa, interactiva y descentralizada, con permanente revisión y realimentación entre iguales (peer review), creando software como bien público.

No debemos de confundir software de código abierto con dos categorías de software muy difundidas:

- *Freeware*, en este caso, el usuario accede gratuitamente al código ejecutable de un producto, pero no al código fuente.
- *Shareware*, se trata de un producto que el usuario puede utilizar durante un determinado periodo de tiempo, período de prueba, y que generalmente, no cuenta con todas las características del producto completo al cual se puede acceder por un determinado precio.

Es oportuno indicar que software libre o de código abierto, no indica que sea gratuito, ya que puede ser necesario pagar ciertos cánones para acceder a él, aunque si bien es cierto que, en general el software libre es gratuito o de bajo coste, ya que su desarrollo es posible gracias a que cientos de programadores pueden aportar su experiencias. Es oportuno destacar que cada día miles de empresas y profesionales se ganan la vida desarrollando e instalando tecnologías libres, y también dando formación y soporte en estas tecnologías, y su número irá indudablemente en aumento. Algunos ejemplos son Cobalt (adquirida por Sun Microsystems), que vende servidores con software Linux altamente personalizado, o empresas como Sharp, que vende el organizador personal de Sharp Zaurus que usa Linux y otros programas libres como base de su software de usuario.

Es necesario tener en cuenta que aun que un software sea gratuito, no tiene por que responder a los criterios de software libre, es decir los cuatro principios básicos de libertad.

2.2 VENTAJAS DEL CÓDIGO ABIERTO

2.2.1 Ventajas económicas

Las ventajas económicas viene derivadas de:

- El usuario no paga por la licencia de uso del programa, sino por el propio programa, ya que puede copiarlo, modificarlo y redistribuirlo.
- El proveedor cobra únicamente por los servicios que presta, en la venta del programa, no por adaptaciones o nuevas versiones
- Normalmente el software asociado a una aplicación de código abierto es también de código abierto. Con el software cerrado suele ser habitual comprar licencias de herramientas que complementen la que ya tenemos.

2.2.2 Independencia del proveedor y mejora de servicios

El cliente es independiente del proveedor ya que, al disponer del código fuente, cualquier programador puede efectuar las modificaciones o adaptaciones. Puesto que el proveedor solo cobra por sus servicios, este concentra todos sus esfuerzos en dar un buen servicio al cliente, ya que es el único modo de mantenerlo.

2.2.3 Disponibilidad de los datos e Integración de los procesos

Estando todo el código disponible, cualquier nuevo desarrollo puede utilizar los datos y procesos del cliente, integrando los distintos programas. Los datos generados siempre serán accesibles, sin obligar para ello al cliente a invertir en licencias. Al conocerse el funcionamiento de los programas, podrán operar entre ellos sin restricciones.

2.2.4 Transparencia y seguridad

Al poderse estudiar el código, las empresas pueden estar seguras respecto al uso que se hace de sus datos y los procesos que se utilizan. No en vano algunos programas han resultado tener puertas traseras o realizar envíos de información sin conocimiento del usuario.

El debate acerca de las virtudes o limitaciones del software de código abierto no sólo se reduce al campo académico, debido a que desmonta las bases del modelo de producción tradicional, sino al ámbito de los negocios, donde las empresas productoras de software propietario perciben el fenómeno como una severa amenaza a sus negocios.

2.3 LICENCIAS

El software de código abierto esta disponible mediante un sistema de licencias que garantizan el libre uso, la copia y la distribución del código fuente.

Una licencia es una autorización formal con carácter contractual que un autor de un software da a un interesado para ejercer "actos de explotación legales". Pueden existir tantas licencias como acuerdos concretos se den entre el autor y el licenciado. Desde el punto de vista del software libre existen distintos grupos de licencias.

Podemos distinguir dos tipos de licencias: licencias permisivas, también llamadas libres o minimalistas, no imponen practicamente ninguna restricción sobre quien recibe el software, dándole permiso de uso, redistribución y modificación; por otra parte, licencias robustas, las cuales si establecen limitaciones sobre quien recibe el software.

2.3.1 Licencias Permisivas

2.3.1.1 Licencia de X Window versión 11 (X11) [3]

Es la licencia usada para la distribución del sistema X Windows, el sistema de ventanas más ampliamente usado en el mundo Unix, y también en entornos GNU/Linux. Es una licencia muy similar a la licencia BSD, que permite redistribución, uso y modificación prácticamente sin restricciones.

2.3.1.2 Zope Public License 2.0 [4]

Esta licencia (habitualmente llamada 'ZPL') es usada para la distribución de Zope (un servidor de aplicaciones) y otros productos relacionados. Es una licencia similar a la BSD, con el interesante detalle de prohibir expresamente el uso de marcas registradas por Zope Corporation.

---

[3] X window system release 11 license. http://www.x.org/Downloads_terms.html.

[4] Zope public license 2.0. http://www.zope.org/Resources/ZPL.

2.3.1.3 Licencia de Apache

Licencia bajo la que se distribuyen la mayor parte de los programas producidos por el proyecto Apache. Es similar a la licencia BSD.

2.3.2 Licencias Robustas

2.3.2.1 Licencia GPL[5]

Una de estas licencias se denomina Licencia Pública General (General Public License) y se conoce por sus siglas GPL o también como copyleft (all rights reversed) y establece que nadie puede restringir la libre circulación del software original y sus modificaciones.

Por lo tanto, el usuario tiene la libertad de utilizar el código, modificarlo, corregirlo, personalizarlo y redistribuirlo, siempre y cuando cumpla con las condiciones que se han fijado para ello. Este tipo de licencia ofrece la seguridad legal de que el software de código abierto se mantendrá disponible para todos y que ninguna persona o empresa se hará propietaria del mismo.

La licencia GPL posibilita la unión de códigos licenciados bajo GLP, es decir si un en mismo programa utilizamos código "A" y código "B" ambos licenciados bajo una licencia GPL, el resultado corresponderá a un nuevo software con código "C" el cual deberá de estar licenciado bajo la licencia GLP.

En la practica esto da lugar a que las licencias del software libre se dividan en dos grandes grupos.

1. Aquellas que pueden ser mezcladas con código licenciado bajo GLP, por lo que inevitablemente se unirán al proceso, ya que el código resultante sea licenciado bajo GLP

2. Aquellas que por ser mezcladas con otros tipos de código que no admiten la licencia

[5] Free Software Foundation. Gnu general public license, version 2, June 1991. http://www.fsf.org/licenses/ gpl.html.

GLP, nunca podrán formar parte del proceso, ya que no cumplen los requisitos necesarios para obtener una licencia GLP.

Existen otras licencias que dan la opción a que el usuario realice modificaciones y que las pueda registrar con licencias privativas y sin revelar el código fuente. Es el caso de la denominada BSD (Berkeley Standard Distribution) Otro tipo de licencia intermedia, entre la recién enunciada y la licencia pública general es la MPL (Mozilla Public License) por la cual se puede registrar un producto derivado del código original pero todo cambio a ese código debe de hacerse público.

2.3.3 Otras licencias

2.3.3.1 Licencia Sleepycat

Es una licencia bajo la cual la empresa Sleeepycat distribuye sus programas. Obliga a ciertas condiciones siempre que se redistribuye el programa o trabajos derivados del programa. En particular, obliga a ofrecer el código fuente, y a que la redistribución imponga al receptor las mismas condiciones. Es muy similar a la GLP aunque mucho menos extensa.

2.3.3.2 eCos License 2.0[6]

Es la licencia bajo la cual se distribuye el eCos, un sistema operativo a tiempo real. Es una modificación de la GPL que no considera que el código que se enlace con programas protegidos por ella queden sujetos a las cláusulas de la GLP si se redistribuyen

2.3.3.3 Affero General Public License[7]

La Licencia Pública General de Affero (Affero General Public License), por sus siglas denominada AGPL, es una licencia derivada de la licencia GPL.

---

[6] ecos license 2.0. http://www.gnu.org/licenses/ecos-license.html.

[7] Affero general public license, 2002. http://www.affero.org/oagpl.html.

Es una interesante modificación de la GPL que considera el caso de los programadores que ofrecen servicios vía web, o en general, vía redes de ordenadores. Este tipo de programas plantean un problema desde el punto de vista de las licencias. Como el uso del programa no implica haberlo recibido mediante una redistribución, aunque el programa este licenciado, alguien puede modificarlo y ofrecer un servicio en la red usándolo, sin redistribuirlo de ninguna forma, y por lo tanto sin estar obligado a redistribuir el código fuente. Por lo que la Affero GPL tiene una cláusula que obliga a que si el programa tiene un medio para proporcionar su código fuente vía web, no se puede desactivar esa característica. Es decir si el autor original incluye esta capacidad en la fuente, cualquier usuario puede obtenerlo, y además esa redistribución esta sometida a las condiciones de la licencia.

2.3.3.4 IBM Public License Version 1.0

Es una licencia que permite la redistribución binaria de trabajos derivados sólo si (entre otras condiciones) se prevé algún mecanismo para que quien reciba el programa pueda recibir su código fuente. La redistribución del código fuente se ha de hacer bajo la misma licencia. Además, esta licencia es interesante debido a que obliga a quien redistribuye el programa con modificaciones, a licenciar automática y gratuitamente las patentes que puedan afectar a esas modificaciones, y que sean propiedad del redistribuidor.

Este tipo de licencia es incompatible con la GPL, ya que contiene restricciones no permitidas por una licencia GPL. Según la FSF (Free Software Foundation) "*esta es una licencia de software libre, pero desafortunadamente tiene una cláusula de elección de la ley que no la hace compatible con la GPL.*" la cual estipula: «este acuerdo se rige por las leyes del Estado de Nueva York y las leyes sobre propiedad intelectual de Estados Unidos». Esta cláusula establece un requisito que va más allá de lo exigido por la GNU GPL, siendo incompatible específicamente con la sección sexta de la GPLv2, sección que prohíbe imponer a los receptores del programa licenciado mayores restricciones en el ejercicio de sus derechos que los contemplados por la antedicha licencia. Por lo tanto como se diferencia de una licencia GPL ya que obliga a licenciar las patentes necesarias para la correcta implementación del software.

2.3.3.5 Licencia BSD (Berkeley Sotware Distribution)

Es una licencia de software libre permisiva, al contrario que la GPL permite el uso del código fuente en software no libre, lo que permite un desarrollo rápido. El autor en esta licencias únicamente mantiene la protección del copyright para posibles reclamaciones de correcta atribución de autoría de la obra o los trabajos derivados de la misma. Por lo tanto el autor, bajo esta licencia, permite la libre redistribución y modificación, sin establecer que se deban mantener las mismas libertades. Bajo este tipo de licencia, se ha utilizado software libre en software privativo como es el caso de Mac OS X.

Muchos autores argumentan que esta licencia asegura el "verdadero" software libre, en el sentido de que el usuario tiene libertad ilimitada sobre el software, pudiendo decidir incluso si distribuirlo o no. Pero existen otras opiniones, las cuales mantienen que este tipo de licencia no contribuye al desarrollo del software libre.

2.3.3.6 Mozilla Public license 1.1[8]

Ejemplo de licencia libre con origen en una empresa. Es una evolución de la primera licencia libre que tuvo el Netscape Navigator, y en su momento fue muy importante por ser la primera vez que una empresa muy conocida decidió distribuir un programa bajo su propia licencia libre.

Hasta ahora hemos dado por supuesto que cada programa se distribuye bajo una única licencia en la que se especificaban las condiciones de uso y redistribución. Sin embargo, un autor puede distribuir obras con distintas licencias.

Para entenderlo, debemos tener en cuenta que cada publicación es una nueva obra y que se puede dar el caso de que se distribuyan versiones que sólo se difieren en la licencia. Como veremos, en la mayoría de los casos, esto se traduce en que dependiendo de lo que el usuario quiera hacer con el software, se encontrará con que tiene que obedecer a una licencia u otra.

Uno de los ejemplos más conocidos de doble licencia es el de la biblioteca Qt, sobre la

[8] Mozilla public license 1.1. http://www.mozilla.org/MPL/MPL-1.1.html.

que se cimienta el entorno de escritorio KDE. Trolltech, una empresa afincada en Noruega distribuía Qt con una licencia propietaria, aunque eximía del pago a los programas que hicieran uso de la misma sin ánimo de lucro. Por esta causa y por sus características técnicas fue elegida a mediados de la década de los noventa por el proyecto KDE, Esto supuso una ardua polémica con la Free Software Foundation, ya que KDE dejaba de ser entonces software libre en su conjunto, al depender de una biblioteca propietaria. Tras un largo debate. Trolltech decidió utilizar el sistema de doble licencia para su producto estrella: los programas bajo GPL podían hacer uso de una versión de Qt GPL, mientras que si se quería integrar con programas con licencias incompatibles con la GPL (por ejemplo, licencias privativas), debían comprarles una licencia especial. Esta solución satisfizo a todas las partes, y hoy día KDE es considerado software libre.

Otros ejemplos conocidos de licencia dual son StarOffice y OpenOffice.org, o Netscape Communicator y Mozilla. En ambos casos el primer producto es propietario, mientras que el segundo es una versión libre.

En la siguiente tabla podemos encontrar los aspectos mas destacado según la familia de licencias:

Familia de licencias	Ejemplos destacados	Características destacadas
Tipo Copyleft	GPL, GFDL	Obligan a la publicación del código fuente cuando se libera código binario; especial preocupación por forzar el efecto copyleft en programas que incluyan código bajo este tipo de licencias
Tipo BSD	Licencia BSD, BSD modificada, Licencia Software Apache	El código licenciado puede mezclarse libremente con código software propietario; algunas especifican obligación de atribuir autoría al proyecto quien generó el código
Tipo MPL	Mozilla Public License	Incompatible con licencias tipo GPL; favorece la adopción de código binario por parte de empresas de software propietario; permite aplicar licencias propietarias a ciertas partes del código, liberando las restantes

Tabla 1: Características según familia de licencias.

Fuente: Cenatic

2.4 DESARROLLADORES

En este punto haremos una breve descripción de los desarrolladores del software libre, ya que sin su colaboración, no seria posible el desarrollo de dicho software. Hace unos quince años podríamos hablar de una colaboración desinteresada por parte de los desarrolladores, pero si bien es cierto, actualmente conocemos que gran parte de las empresas poseen trabajadores desarrollando software libre en horas de trabajo. Hann. Et al, (2004) comprobaron empíricamente para Committers Apache Software Foundation, la existencia de desarrolladores de software de código abierto, los cuales poseen una remuneración mayor, concluyendo que esto se debe a que dichos desarrolladores poseen gran poder dentro de la empresa, ya que su trabajo, conlleva con sigo una mejora de sus conocimientos, y al tratarse de software libre estos podrían ser puestos rápidamente en práctica por empresas de la competencia

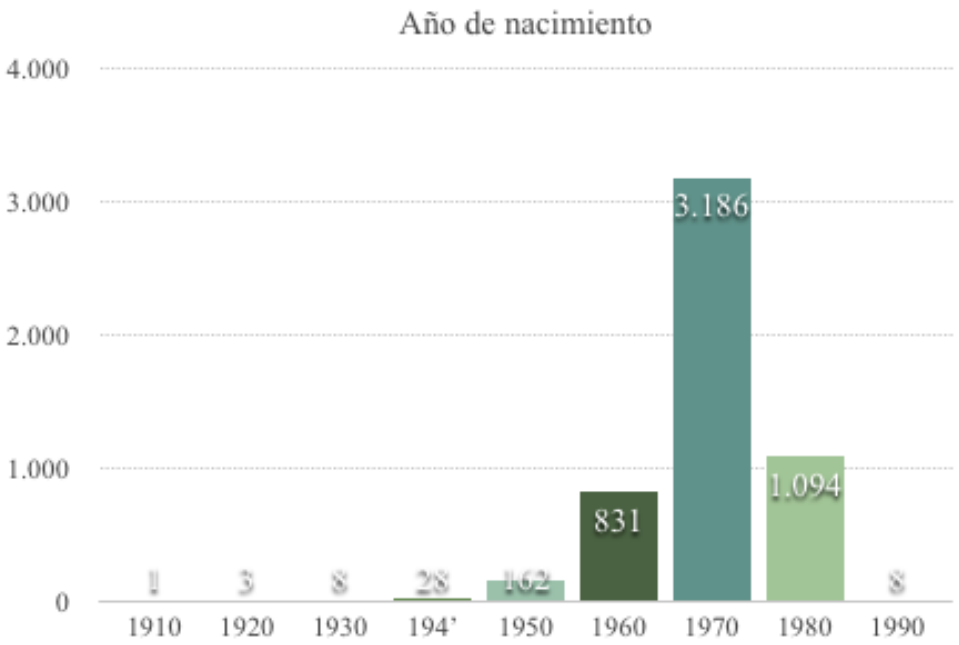
Desde hace pocos años, se ha venido realizando un gran esfuerzo para conocer a las comunidades que hacen posible la creación de un software libre, anteriormente se las mitificaba bajo la cultura del *hacker.* Algunos científicos consideran que conocer quién se implica y por qué puede ser de gran utilidad para la generación de software libre.

Ser un hacker es muy divertido, pero es un tipo de diversión que supone mucho esfuerzo. El esfuerzo supone motivación. Los deportistas de éxito obtienen su motivación de un tipo de placer físico en hacer que sus cuerpos funcionen, en llevarse a si mismos más allá de sus propios límites físicos. De forma similar, para ser un hacker tienes que experimentar una emoción básica al resolver problemas, al afilar tus aptitudes y al ejercitar tu inteligencia.

Raymond (2007)

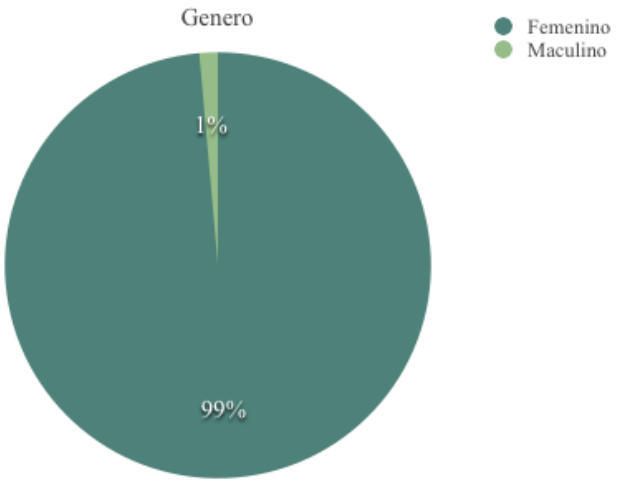
A continuación se mostraran una serie de datos relativos a las características más comunes de los desarrolladores. Basados en el estudio realizado por Robles, Scheider, Tretkowski y Niels (2001) del cual se establece que la edad media de los desarrolladores esta comprendida entre 20 y 24 años, de donde se deduce que en su mayoría se encuentran en la edad universitaria, lo que no es de extrañar ya que el

mundo universitario esta íntimamente ligado al desarrollo de software libre. Respecto a su género, podemos afirmar que en su mayor parte son hombres (98,6%).



Gráfica 1: Año de nacimiento de los desarrolladores.

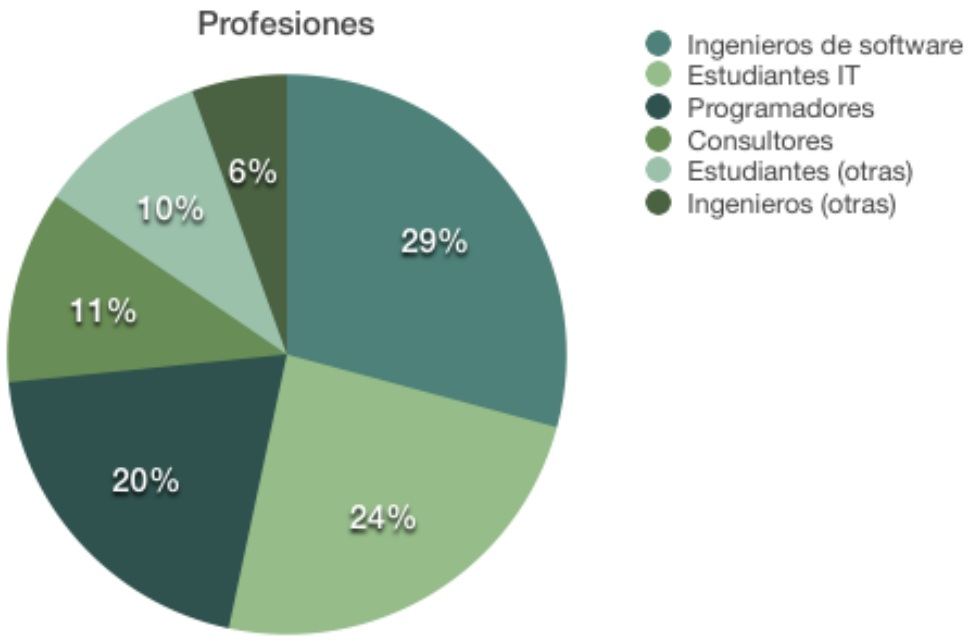
Fuente: Robles, Scheider, Tretkowski y Niels (2001)



Gráfica 2:Genero de los desarrolladores.

Fuente: Robles, Scheider, Tretkowski y Niels (2001)

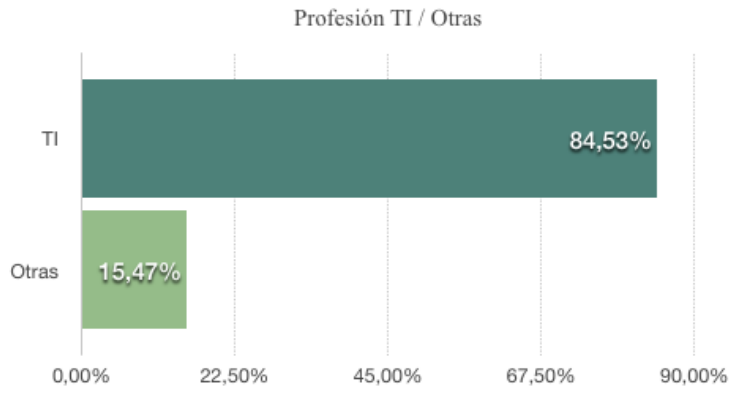
Profesionalmente los desarrolladores se definen como ingenieros de software, estudiantes, programadores, consultores o profesores de universidad. También se ha podido observar una gran interdisciplinar, lo que genera una gran riqueza en cuanto a interés, procedencia en la composición de los equipos de desarrollo, lo que nos lleva a preguntarnos, si de no ser este el modelo de trabajo, sería posible, en una industria moderna, llegar a alcanzar este nivel de heterogeneidad, que por su puesto le confiere al software libre unas características diferentes sobre cualquier otro tipo de proyecto.



Gráfica 3: Profesiones de los desarrolladores.

Fuente: Robles, Scheider, Tretkowski y Niels (2001)

En el gráfico siguiente podemos observar que la mayor parte de los programadores esta relacionado con el mundo de las Tecnologías de la Información (IT)



Gráfica 4: Profesiones TI/Otras.

Fuente: Robles, Scheider, Tretkowski y Niels (2001)

Un punto muy importante es las motivaciones que hacen posible que el software libre exista, ya que nos ayuda a entender porque existe algo, que según la teoría económica sería inviable. Basándonos en las recopilaciones recogidas por J. González, Seoane y Robles (2003), podemos observar que las motivaciones de los desarrolladores se dividen en cinco:

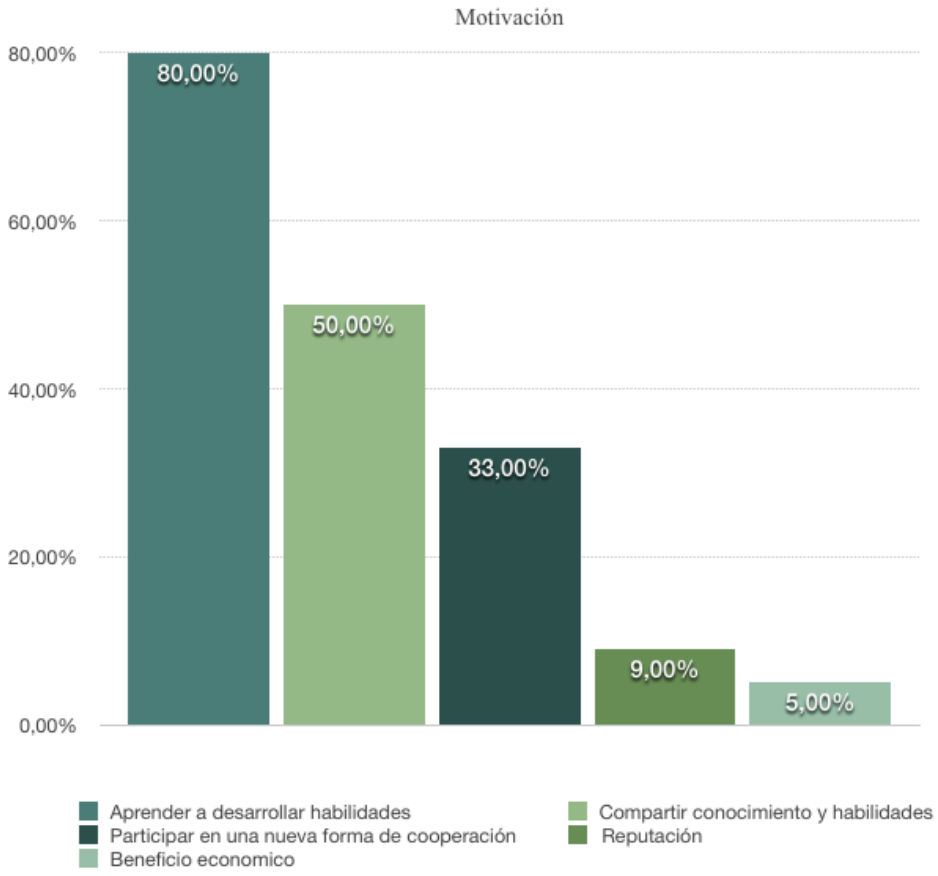
Aprender a desarrollar nuevas habilidades, (80%) este dato no nos ha de extrañar, ya que la mayor parte se encuentran cursando estudios universitarios, y es sabido que una persona más preparada y con mayores habilidades es mas cotizada. Es decir la mayor parte de los autores, cuando desarrolla uno de estos software, lo hace por el mero hecho de poner en práctica sus habilidades.

Compartir conocimientos y habilidades, (50%) esta motivación, en mi opinión viene ligada a la anterior, ya que si además de poner en práctica nuestras habilidades, las podemos publicar, a vista de otros compañeros del mismo campo, estos podrán dar su opinión, y nos ayudaran a mejorar y a aprender nuevos conceptos, ya que si queremos aprender una de las mejores formas es enseñar lo que uno sabe, y a si comenzar una retroalimentación.

Participar en una nueva forma de cooperación, (33%) los desarrolladores se muestran entusiasmados por su nuevo movimiento, por lo que no es de extrañar que otros muchos se quieran unir a este movimiento. Simplemente lo podríamos considerar una nueva moda en la que muchos están interesados en colaborar, para ellos es una forma de hacer historia con su trabajo.

Reputación, (9%). Dentro de las propias comunidades de desarrolladores, los autores pueden ganarse cierta reputación que contribuya positivamente a la hora de encontrar un puesto de trabajo dentro de su ámbito profesional, lo que llevaría indudablemente al punto siguiente

Beneficio económico (5%). Algunos desarrolladores consideran que la realización de software libre es una inversión de futuro, ya que el desarrollo del mismo les puede reportar nuevas habilidades, que formaran una mejor reputación lo que les permitirá el acceso a un mejor puesto de trabajo, lo que indudablemente conllevara a un mayor beneficio económico.



Gráfica 5: Motivaciones de los desarrolladores.

Fuente: Robles, Scheider, Tretkowski y Niels (2001)

A continuación se expone una tabla identificando los principales grupos de interés más influyentes en las actividades de una organización desarrolladora de software libre.

Categoría	Stakeholder
Internos	Propietarios de la empresa o socios y miembros de la organización
	Alta dirección de la organización y liderazgo u órganos de gobierno del proyecto
	Empleados de la organización
Externos que financian o participan en el proceso de desarrollo	Agentes que incorporan el desarrollo a sus productos de software o hardware
	Colaboradores independientes ajenos a la organización
	Aliados en el desarrollo tecnológico y Universidades y grupos de investigación
	Instituciones que apoyan o financian el desarrollo
	Donantes en general
Externos ajenos a la financiación o proceso de desarrollo	Clientes y usuarios finales del desarrollo
	Comunidades de otros desarrollos de código abierto
	Competidores (Software libre o privativo)
	Administraciones públicas
	Sociedad en general

Tabla 2: Grupos de interés, Stakeholdes.

Fuente: García-García & Alonso de Magdaleno, 2013

2.5 MODELO DE FINANCIACIÓN DEL SOFTWARE DE CÓDIGO LIBRE

A diferencia del software propietario, que viene con licencias restrictivas de derecho de autor, el software de código abierto puede ser entregado de forma gratuita. Esto significa que sus creadores no pueden exigir a cada usuario pagar un canon para financiar el desarrollo. Por lo que es su lugar han surgido una serie de modelos alternativos para la financiación de su desarrollo.

2.5.1 Financiación pública

La entidad financiadora puede ser directamente un gobierno (local, regional, nacional o incluso supranacional) o una institución pública (por ejemplo, una fundación). En estos casos, la financiación suele ser similar a la de los proyectos de investigación y desarrollo, y de hecho es muy habitual que provenga de entidades públicas promotoras de I+D. Normalmente, la entidad financiadora no busca recuperar la inversión (o al menos no de forma directa), aunque suele tener objetivos claros (favorecer la creación de tejido industrial e investigador, promover cierta tecnología o cierto tipo de aplicaciones, etc.).

En la mayor parte de estos casos, no se encuentra explícitamente la financiación de productos o servicios relacionados con software libre, sino que suelen ser el subproducto de un contrato con otros fines más generales. Por ejemplo, la Comisión Europea, dentro de sus programas de investigación, financia proyectos orientados a mejorar la competitividad europea en ciertas áreas. Algunos de estos proyectos tienen como parte de sus objetivos usar, mejorar y crear software libre en su ámbito de investigación (como herramienta para la investigación, o como producto derivado de ella).

Las motivaciones para este tipo de financiación son muy variadas, pero pueden destacarse las siguientes

- Científica. Este es el caso más habitual en proyectos de investigación financiados con fondos públicos. Aunque su objetivo no es producir software, sino investigar en un determinado campo (relacionado o no con la informática), es muy posible que para ello sea preciso desarrollar programas que se usen como herramientas necesarias para alcanzar las metas del proyecto. Normalmente el proyecto no está interesado en comercializar estas herramientas, o incluso está activamente

interesado en que otros grupos las utilicen y mejoren. En estos casos, es bastante habitual distribuirlos como software libre. En este caso, los recursos que consiguió el grupo que realiza la investigación se dedicaron en parte a la producción de este software, por lo que se puede decir que fue desarrollado con financiación pública.

- Promoción de estándares. Tener una implementación de referencia es una de las mejores formas de promover un estándar. En muchos casos eso supone tener programas que formen parte de esa implementación. Para que la implementación de referencia sea de utilidad en la promoción del estándar, es preciso que esté disponible, al menos para comprobar interoperabilidad, para todos los que quieran desarrollar productos que se acojan a ese estándar. Y en muchos casos, es conveniente también que los fabricantes puedan directamente adaptar la implementación de referencia para usarla en sus productos, si así lo desean. De esta manera se desarrollaron, por ejemplo, muchos de los protocolos de Internet que hoy se han convertido en norma universal. En estos casos, la liberación de esas implementaciones de referencia como software libre puede ayudar mucho en esa promoción. De nuevo, en estos casos el software libre es un subproducto, en este caso de la promoción del estándar.

- Social. El software libre es una herramienta de gran interés en la creación de la infraestructura básica para la sociedad de la información. Las entidades interesadas en utilizar software libre para ayudar al acceso universal a esta sociedad de la información pueden financiar proyectos relacionados con el software libre, normalmente proyectos de desarrollo de nuevas aplicaciones o de adaptación de otras existentes.

2.5.2 Financiación privada sin ánimo de lucro

Este es un tipo de financiación con muchas características similares a las del caso anterior, donde la financiación la realizan normalmente fundaciones u organizaciones no gubernamentales. La motivación directa en estos casos suele ser producir software libre para su uso en algún ámbito que la entidad financiadora considera especialmente relevante, pero también puede encontrarse la motivación indirecta de contribuir a resolver un problema.

En general, tanto los motivos para realizar esta financiación como sus mecanismos son

muy similares a los de financiación pública, aunque naturalmente están siempre marcados por los objetivos de la entidad que financia.

2.5.3 Financiación por quien necesita mejoras

Este tipo de financiación para el desarrollo de software libre, ya no tan altruista, es el que tiene lugar cuando alguien necesita mejoras en un producto libre. Por ejemplo, una empresa, para uso interno, necesita de cierta funcionalidad en un programa dado. O bien necesita que ciertos errores en un programa sean corregidos. En este tipo de casos, lo habitual es que la empresa en cuestión contrate el desarrollo que necesita. Este desarrollo, muy habitualmente (bien porque lo impone la licencia del programa modificado, bien porque la empresa lo decide así) es software libre.

2.5.4 Financiación con beneficios relacionados

En este caso, lo que busca la entidad financiadora es conseguir beneficios en productos relacionados con el programa a cuyo desarrollo aporta recursos.

Algunos ejemplos de productos relacionados con un software dado son:

- Libros. La empresa en cuestión vende manuales, guías de uso, textos para cursos, etc. relacionados con el programa libre que ayuda a financiar. Por supuesto otras empresas pueden vender también libros relacionados, pero normalmente el financiar el proyecto le dará acceso antes que a sus competidores a desarrolladores clave, o simplemente conseguirá con ello una buena imagen de cara a la comunidad de usuarios del programa en cuestión.

- Hardware. Si una empresa financia el desarrollo de sistemas libres para cierto tipo de hardware puede dedicarse a vender con más facilidad ese tipo de hardware. De nuevo, al ser libre el software desarrollado, pueden aparecer competidores que vendan aparatos del mismo tipo, usando esos desarrollos, pero sin colaborar a su financiación. Pero incluso así, la empresa en cuestión tiene varias ventajas sobre sus competidores, y una de ellas puede ser que su posición como aportadora de recursos para el proyecto le permitía influir para conseguir que los desarrollos que se realicen con más prioridad sean los que más le interesen.

- CDs con programas. Probablemente el modelo de este tipo más conocido es el de las empresas que financian ciertos desarrollos que luego aplican a su distribución de software. Por ejemplo, tener un buen entorno de escritorio puede ayudar mucho a vender CDs con una cierta distribución de GNU/Linux, y por tanto, financiar su desarrollo puede ser un buen negocio para quien los vende.

Es importante darse cuenta de que, para estar en este apartado, la financiación en cuestión ha de hacerse con ánimo de lucro, y para ello la entidad financiadora ha de percibir algún beneficio posible en esta financiación. En los casos reales, sin embargo, es habitual que haya siempre una combinación de ánimo de lucro y altruismo cuando una empresa aporta recursos para que se realice un programa libre del cual espera beneficiarse indirectamente.

2.5.5 Financiación como inversión interna

Hay empresas que, directamente como parte de su modelo de negocio, desarrollan software libre. Por ejemplo, una empresa puede decidir fabricar un nuevo producto el cual lleve instalado dicho software libre. La empresa obtendrá sus beneficios de la venta de dicho dispositivo, pero para que el dispositivo funcione correctamente, habrá invertido en el desarrollo de nuevo software libre.

Este modelo podría considerarse una variante del anterior (financiación indirecta), siendo los 'beneficios relacionados', las ventajas que obtenga la empresa de la producción del programa libre. Pero por ser en este caso el producto libre en sí mismo el que genera que produzca los beneficios, es conveniente abrir una clasificación específica para estos casos, como es el caso de la financiación como inversión interna.

2.5.6 Otros modelos de financiación

En la búsqueda de financiación para el desarrollo de software libre, existen una gran variedad de métodos que citaremos a continuación, estos se conocen con el término de financiación masiva o crowdfunding. Según Mollick (2013) el crowdfunding consiste en una cooperación colectiva llevada a cabo por personas que realizan una red para conseguir dinero u otros recursos. Se suele utilizar Internet como medio para financiar esfuerzos e iniciativas de otras personas u organizaciones. Los beneficios obtenidos una vez finalizado el software, serán objeto de beneficio de toda la comunidad inversora (Belleflamme, Lambert, Schwienbacher, 2013)

Utilización de mercados para poner en contacto a desarrolladores y clientes. La idea que sostiene este modo de financiación es que, sobre todo para pequeños desarrollos, es difícil que un cliente que lo desea pueda entrar en contacto con un desarrollador que pueda acometerlo de forma eficiente. Para mejorar esta situación, se postulan los mercados de desarrollo de software libre, donde los desarrolladores publicarían sus habilidades y los clientes los desarrollos que precisan.

Venta de bonos para financiar un proyecto. Esta idea de financiación es similar a la de los mercados de bonos a los que acuden las empresas, pero orientado al desarrollo de software libre. Tiene unas cuantas variantes, pero una de las más conocidas funciona así: Cuando un desarrollador (individual o empresa) tiene idea de un nuevo programa o una mejora a uno existente, lo escribe en forma de especificación, estima el coste que tendría su desarrollo, y emite bonos para su construcción. Estos bonos tienen un valor que se ejecuta sólo si el proyecto se termina finalmente. Cuando el desarrollador ha vendido suficientes bonos, comienza el desarrollo, que va financiando con préstamos basados en ellos. Cuando termina el desarrollo, y se certifica por alguna tercera parte independiente que efectivamente lo realizado cumple las especificaciones, el desarrollador 'ejecuta' los bonos que había vendido, paga sus deudas, y lo que le queda son los beneficios que obtiene por el desarrollo. ¿Quién estaría interesado en adquirir los mencionados bonos? Obviamente, los usuarios que desearan que ese nuevo programa, o esa mejora a uno ya existente, se realizaran. De alguna manera, este sistema de bonos permitiría que las partes interesadas fijaran (siquiera parcialmente) las prioridades de los desarrolladores mediante la compra de bonos. Esto permitiría también que no hiciese falta que una sola entidad asumiese los costes de desarrollo, sino que podrían repartirse entre muchas (incluyendo individuos), que además sólo tendrían que pagar si finalmente el proyecto termina con éxito.

Cooperativas de desarrolladores. En este caso, los desarrolladores de software libre, en lugar de trabajar individualmente o para una empresa, se reúnen en algún tipo de asociación. Su funcionamiento es similar al de una empresa, matizado quizás por su compromiso ético con el software libre, que puede ser parte de sus estatutos (aunque también una empresa puede hacer esto). En este tipo de organizaciones pueden darse combinaciones variadas de trabajo voluntario con trabajo remunerado. Un ejemplo de este tipo de organización es Free Developers.

Sistema de donaciones. Consiste en habilitar un mecanismo de pago al autor de un determinado software en la página *web* que alberga el proyecto. De esta forma, los usuarios interesados en que dicho proyecto continúe publicando nuevas versiones pueden apoyarlo económicamente, realizando donaciones voluntarias a modo de financiación para el desarrollador.

Finalmente podemos destacar que el nuevo sistema ofrece un producto que potencialmente es más flexible, más efectivo, de menor costo e independiente de que el proveedor realice correcciones o lance nuevas versiones.

Es ineludible que la popularidad del software libre es creciente y que ya es una alternativa muy a tener en cuenta por muchos sectores. Las ventajas que toda una comunidad de desarrolladores pueden proporcionar no son fáciles de ignorar, y eso hace que cada vez más empresas se decidan por pasarse a los programas libres. En el apartado siguiente, se mostraran algunos ejemplos que pueden verificar esté crecimiento.

2.6 PRESENCIA EN EL MUNDO

Cuando en 1984 Richard Stallman decidió fundar la FSF (Free Software Foundation) quizás no imaginaba la revolución que estaba forjando con ello. Lo cierto es que el proyecto GNU de Stallman pretendía recuperar el espíritu cooperativo que imperaba en los inicios de la era computacional y que la fiebre de las patentes y el software propietario habían enterrado.

Para construir un mundo de software libre, lo primero y más básico es tener un sistema operativo libre. Para principios de los 90, y combinando la estructura de GNU con el núcleo Linux, consiguió su objetivo y dio paso a una nueva generación de sistemas GNU basados en Linux, como Red Hat o Debian.

En sus primeros años de formación, GNU y la comunidad de software libre en general, no suponían una amenaza seria para las poderosas empresas de software propietario. Sin embargo, la popularidad de este tipo de programas sin restricciones de modificación o distribución comenzó a extenderse. Miles de aficionados a la informática se subieron al carro del software libre y se creó una comunidad de usuarios y desarrolladores cada vez mayor, hasta hoy en día, que cuenta con millones de personas repartidas por todo el mundo.

Pese a menospreciarla en un principio, Microsoft se ha dado cuenta que la comunidad del software libre es ya una amenaza a su privilegiada posición en el mercado. Esta amenaza se pone cada vez más de manifiesto. Los números en 2007- indicaban que Microsoft no tenía rival. El 94,28% de los ordenadores tienen Microsoft Windows y el 67,6% utiliza Microsoft Internet Explorer. Pero este mismo dato en el 2014 ha variado significativamente, siendo el 61,72% los ordenadores que poseen Microsoft Windows y el 18,5% utilizan Microsoft Internet Explorer, podemos observar que en un periodo de siete años Microsoft ha experimentado una caída respecto a su presencia en el mundo de los ordenadores.

Es indudable que la popularidad del software libre es creciente y que ya es una alternativa muy a tener en cuenta por muchos sectores que no hace muchos tiempo desconocían su existencia. Las ventajas que toda una comunidad de desarrolladores pueden proporcionar no son fáciles de ignorar. El uso del software libre es especialmente popular en Sudamérica, dónde existe un elevado número de usuarios y

algunos países promocionan el uso de este tipo de programas, subvencionándolos o proporcionándoselos a los organismos estatales. De la misma manera pasa en otros lugares del mundo, incluido España, dónde el ejemplo más claro lo tenemos en Extremadura[9], cuyo gobierno autonómico promueve su versión de Linux (Linex) e incorporándola en los ámbitos educacionales, sanitarios, gubernamentales… También Guadalinex[10] en Andalucía. Los gobiernos se han dado cuenta de que el software libre es una piedra angular en las Tecnologías de la Información, y están incluyendo su promoción como parte de sus programas electorales. Sin embargo, hay un sector en el que el software libre, con Linux a la cabeza, no está siendo capaz de penetrar: el usuario medio. Los habituales problemas de compatibilidad de aplicaciones o hardware, el interfaz menos atractivo que Windows, y la curva de aprendizaje marcadamente más pronunciada que el programa de Microsoft, está imposibilitando a Linux hacerse un hueco en los ordenadores del usuario medio. Stallman y las demás cabezas pensantes de las comunidades son conscientes de ello y prevén que en dos o tres años, Linux estará en posición de luchar mano a mano con Windows por los ordenadores del mundo.

En definitiva según datos del Instituto Nacional de Estadística, más del 75% de las empresas españolas y el 90% de las administraciones publicas utilizan ya el software libre[11].

[9] http://linex.gobex.es/
[10] http://www.guadalinex.org/
[11] ABC, septiembre 2011,El 75% de las empresas y el 90% de las administraciones tienen software libre. Disponible en:www.abc.es/agencias/noticias.asp?noticia=935217

2.7 RESPONSABILIDAD SOCIAL Y SOFTWARE LIBRE

En un primer momento, la responsabilidad social se consideraba tan sólo como la responsabilidad de la empresa de proporcionar un retorno financiero máximo para sus accionistas. Fue en 1960 cuando Keith Davis sugirió que la responsabilidad social se refiere "*Las decisiones y medidas adoptadas se deben a razones de, al menos parcialmente más allá de el interés económico o técnico*" junto con él, Eells y Walton (1961) argumentaron que la responsabilidad social empresarial se refiere a los "*problemas que surgen cuando la empresa corporativa proyecta su sombra en la escena social y los principios éticos que deben regir la relación entre la sociedad anónima y la sociedad*". La cuestión seguía siendo la conciliación de la orientación económica de la empresa con su orientación social. En este punto de vista, una conceptualización de cuatro partes de la Responsabilidad Social Corporativa (RSC) incluyen la idea de que la empresa no sólo tienen obligaciones económicas y legales, como en un principio se creía, sino que posee unas responsabilidades éticas y filantrópicas (Carroll 1979). Estas cuatro categorías podrán ser representadas como una pirámide.

Componentes Económicos	Componentes Legales
Maximización de las ganancias **Rentabilidad máxima** **Mantener una fuerte posición competitiva** **Alto nivel de eficiencia operativa**	Cumplir con las expectativas del gobierno y la ley Necesario reconocer la firma como respetuosa con sus obligaciones legales Proporcionar bienes o servicios que al menos cumplan con el mínimo legal de requisitos
Componentes Éticos	**Componentes Filantrópicos**
Cumplir con las expectativas de la sociedad y las costumbres **Conocer y respetar las normas morales y éticas y su evolución** **Reconocer que la integridad y el comportamiento ético van más allá del mero cumplimiento de las leyes y reglamentos**	Cumplir las expectativas de caridad social. Participación de los gerentes en voluntariados u actividades de caridad dentro de la localidad. Prestar asistencia a las instituciones educativas privadas y públicas. Importante ayudar en la mejora de la "Calidad de vida."

Tabla 3: Componentes de la Responsabilidad Social Corporativa

Fuente: The Pyramid of Corporate Social Responsibility: Toward the Moral Management of Organizational Stakeholders (1991)

En un mundo donde gran parte de la innovación se encuentra en el software, el software libre puede ser considerado como un modelo de transferencia tecnológica sin fricciones (Lippoldt & Stryszowski, 2009). Esta cualidad le ha conferido la denominación de modelo de innovación privado-colectivo (Von Hippel & Von Krogh, 2003) ya que como hemos estudiado anteriormente, este tipo de software es financiado por agentes privados pero produce un beneficio colectivo.

Analizando los cuatro componentes de la responsabilidad social en el uso de software libre, podemos observar como el software libre responde a un modelo por el cual se genera valor a la sociedad en su conjunto.

Es de entender que tanto los componentes económicos como legales se dan en el uso del software libre, ya que cualquier empresa que utilice como base de su negocio el software libre, tratara de maximizar con ellos sus beneficios, así mismo responderá ante las leyes. Pero observemos que ocurre con las componentes éticas, con el uso del software libre, respondemos a las normar morales de una sociedad, la cual esta basada en las cuatro libertades del software libre, pero además responde a las componentes filantrópicas ya que cualquier modificación existente en dicho software ha de ser comunicada a toda la comunidad.

Después de lo estudiado anteriormente, *al menos un 80% del sector de desarrollo de software emplea algún producto de software libre (Drive 2010) y que su contribución a la economía de la Unión Europea es de 450 millones de euros (Daffara, 2012.)*, no podemos poner en duda que el software libre genera valor para la sociedad en conjunto. Pero ninguna empresa ni entidad no lucrativa dedicada a la creación de software libre emplea este concepto como estrategia de comunicación, ni tampoco, los estándares de RSC han creado un conjunto de indicadores que puedan ser de aplicación.

Desde el punto de vista de la sostenibilidad social, la RSC entiende a la empresa como un conjunto de individuos que persiguen determinados objetivos, de forma que el valor generado en la persecución y obtención de esos objetivos, aporta una mejora de la organización, la comunidad y la sociedad. No podemos dudar que esta definición refleja el proceso de desarrollo del software libre.

El compromiso corporativo del software libre significa compartir tecnología y recursos con las comunidades de todo el mundo, favoreciendo el acceso a las TIC y creando nuevas oportunidades de desarrollo. El modelo de desarrollo del software libre, su capacidad de transferir la innovación tecnológica y de generar desarrollos colectivos, han creado un producto libre, a disposición de cualquiera, ya sea una gran empresa o un pequeño emprendedor. Esto debería de ser reflejado en los informes de RSC de las distintas organizaciones que trabajan con desarrollos de código abierto, pero no es frecuente encontrar ninguna referencia al respecto en los informes anuales en las empresas que lo desarrollan tanto para uso interno como externo. Esto puede deberse, en gran parte a dos conceptos:

- La dificultad de contabilizar el coste invertido, ya que su desarrollo colectivo y colaborativo impide una correcta estimación.
- El marco contable aceptado a nivel global impide que el valor creado por el software libre quede reflejado de manera clara y objetiva en los informes contables (García-García & Alonso de Magdaleno, 2013).

En la actualidad, comunidades, fundaciones o compañías como Mozilla, Linux o Red Hat destacan en sus informes las contribuciones externas con el fin de apoyar el desarrollo de sus proyectos futuros. En paralelo, se han desarrollado protocolos para evaluar los procesos de desarrollo centrándose en aspectos como la madurez, supervivencia o estrategia de las organizaciones que gestionan el proyecto, incorporando también aspectos funcionales y operativos. A continuación se destacan algunos ejemplos de estos protocolos: el modelo Open Source Maturity (OMM) de Capgemini, Navica o QualiPSo, o los modelos Qualification and Selection of Open Source Software (QSOS), Open Business Readiness Rating (OpenBRR), Open Business Quality Rating (OpenBQR) y Model for Open Source Software Trustworthiness (MOSST) de QualiPSo. Aunque estos protocolos son de gran importancia, no son suficientemente correctos para evaluar el valor económico y social del software libre, es necesario incluir sistemas de reporte de la RSC.

En un estudio actual sobre la comunicación de la RSC García-García & Alonso de Magdaleno, (2013) se han tratado de identificar los principales grupos de interés o indicadores que puedan servir como base para la formulación de informes de RSC . En definitiva las entidades interesadas en la implementación y el mantenimiento del

software libre deben de generar sinergias entre una comunicación afectada por su impacto social y sus políticas de calidad. Puntos a tener en cuenta para generar dicha sinergia, se han establecido en el estudio anteriormente mencionado, destacando las siguientes consideraciones:

- Alinear el proceso de evaluación de la RSC con los procesos de mejora de calidad de los desarrollos de software libre y su proceso de vida (ISO/IEC 12207).
- Evaluación de calidad y madurez del desarrollo (ISO/IEC 15504).
- Procesos de provisión eficaz de servicios TIC a la organización o a terceros (ISO/IEC 20000-1).
- Código de buenas prácticas para el sector (ISO/IEC 20000-2)
- También el uso de procesos más generales de gestión de la calidad, no específicos del sector TIC (ISO 9001) es una oportunidad a explotar por el sector de desarrollo de software libre o por aquellas entidades que liberan su software de uso interno.

2.8 SOFTWARE LIBRE EN LAS ADMINISTRACIONES PÚBLICAS

En los últimos años diversas administraciones públicas de todo el mundo han iniciado un proceso de migración del software propietario al software libre. Según ALIAL (2010) "*Una de las principales barreras iniciales para la adopción de Software Libre por parte de las Administraciones es la necesidad de definir las condiciones generales que deben cumplir los proveedores a la hora de desarrollar soluciones para ellos*."

El proceso de migración, que Europa está promoviendo por la Comisión Europea desde el plan eEurope 2002, ha ido alcanzando paulatinamente un umbral significativo. Sin ir mas lejos, en nuestro país, podemos destacar una serie de casos de interés:

- En 1999 el Ministerio de Administraciones Públicas desarrolló sobre servidores GNU/Linux, los servicios proporcionado por las Delegaciones del Gobierno. Estos se encargaban del almacenamiento permanente de datos, la conectividad, la distribución de software, el correo electrónico y el acceso a la intranet/Internet. En 2001 el Ministerio de Justicia se unió a la iniciativa migrando a servidores GNU/Linux las sedes judiciales de su competencia.
- La Junta de Extremadura puso en marcha diversas iniciativas basadas en el software libre, estos proyectos fueron implantados en su mayoría con su propia variante del sistema operativo GNU/Linux, denominado gnuLinEx. Este sistema operativo ha recibido varios premios que refrendan su éxito, incluyendo el Premio Europeo de la Innovación Regional otorgado por la Comisión Europea. Seguidamente fue la Junta de Andalucía, la que tomando como base el sistema operativo desarrollado en Extremadura, creó su propia distribución denominada Guadalinex.
- Junto a estos primeros ejemplos tenemos otras iniciativas, menos conocidas, llevadas a cabo por diversas Comunidades Autónomas y Ayuntamientos desde el año 2003. Podemos destacar: la Comunidad de Madrid, País Vasco, Comunidad Valenciana, Navarra y Galicia. En los Ayuntamientos, muchas de las iniciativas de adopción de software de fuentes abiertas se centran en la migración de algunas de sus infraestructuras de comunicación. Entre ellas destacan, claramente, los servidores web institucionales

A pesar de ello, uno de los principales problemas radica en que cuando una empresa desarrolla un software para un Ayuntamiento, esta le oferta la licencia a un precio determinado, pero dicha empresa seguirá ofreciendo el mismo producto a distintos ayuntamientos, por lo que la Administración esta desembolsando un dinero en diferentes puntos para obtener el mismo servicio. Si la Administración trata de negociar con la empresa, para obtener las distintas licencias por el valor de una de ellas, la empresa puede alegar que este es su negocio, y si le ofreciera el mismo servicio para todos los puntos el precio no sería el mismo.

2.8.1 Ventajas del uso de software libre en las Administraciones Públicas

Algunas de las ventajas del uso de software libre en las Administraciones Públicas, son los siguientes:

- Fomento de la industria local, las empresas locales podrían competir ofreciendo servicios a las administraciones, lo que les confiere de ciertas ventajas frente a otras empresas, ya que pueden aprovechar sus ventajas competitivas: mejor conocimiento de las necesidades del cliente, cercanía geográfica, etc

- Independencia de proveedor y competencia en el mercado, por norma general una organización prefiere depender de un mercado en régimen de competencia que de un sólo proveedor, ya que este puede poner condiciones demasiado restrictivas. Pero en el caso de las Administraciones Públicas está preferencia se convierte en un requisito, ya que no puede, por lo general, elegir contratar a un suministrador, sino debe de especificar sus necesidades de forma que cualquier empresa interesada pueda cubrirlas. Por lo que si dispone de un software libre, para cualquier empresa interesada sería fácil proporcionar las necesidades especificadas. Incluso, si la Administración Pública en un futuro desea cambiar de proveedor podría hacerlo, ya que el producto seguirá siendo el mismo.

- Flexibilidad y adaptación a las necesidades exactas, aunque la adaptación a las necesidades exactas es algo necesario para cualquier organización, en el caso de las Administraciones Públicas es un factor muy importante en el éxito de la implantación del sistema informático. En el caso de disponer de un software propietario, para llegar a adaptar este a sus necesidades han de llegar aun

acuerdo, pero si disponen de un software libre, cualquier empresa podría realizar las modificaciones necesarias para adaptar el software a las especificaciones deseadas por la administración.

- Escrutinio público de seguridad, para las Administraciones Públicas poder garantizar que sus sistemas hacen sólo lo que esta previsto que hagan es un requisito fundamental, incluso en muchos países es un requisito legal. En el caso de usar un software propietario, sin acceso al código fuente, es difícil de demostrar que cumplan esta condición, pero si hablamos de un software libre, es fácil comprobar lo que realmente puede hacer ese software.

- Disponibilidad a largo plazo, es difícil asegurar que un software propietario estará disponible cuando hayan pasado largos periodos de tiempo, y aun peor sería conseguir que funcione en una plataforma más actual, para conseguir su disponibilidad deberíamos de llegar a grandes acuerdos económicos, pero en el caso de un software libre, es fácil que cualquier programador acceda a su código fuente y actualice el software.

2.8.2 Dificultades de adaptación

- Desconocimiento, el principal problema del software libre, es que es un gran desconocido para aquellos que toman las decisiones, afortunadamente este problema se esta disminuyendo paulatinamente.

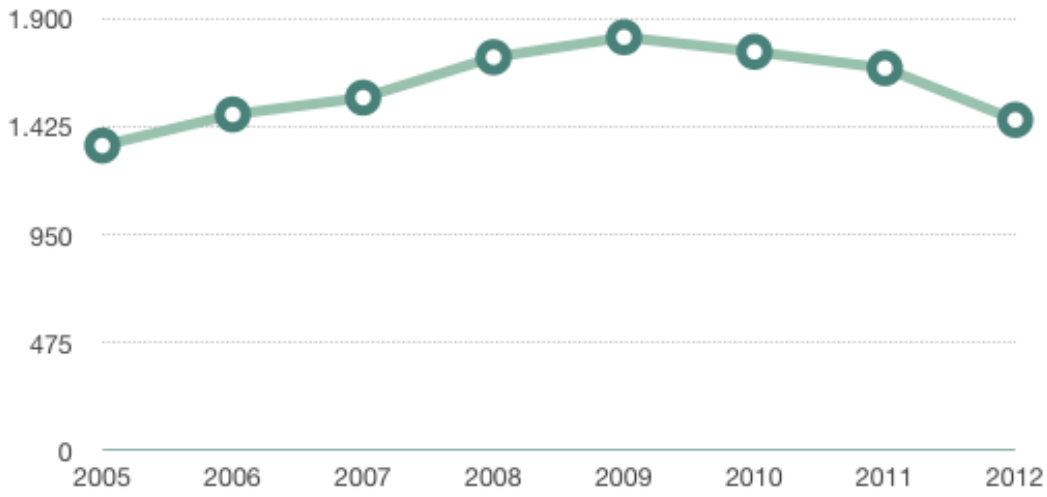
- Falta de adecuación de los servicios de contratación, generalmente los mecanismos de contratación paran por la decisión de partidas de gasto, destinadas a la compra de productos informáticos, pero si se desea trabajar con software libre, la compra del software posee un desembolso insignificante, pero si es necesaria la inversión en servicios a su alrededor.

- Falta de estrategia de implantación, en muchos de los casos el software libre comienza a usarse por su bajo coste, lo que causa que la mayor parte de las ventajas del software libre se pierda por el camino.

2.8.3 Datos cuantitativos

La visión cuantitativa más completa, que puede conseguirse sobre la situación actual del uso del software libre en las Administraciones Públicas españolas, se extrae del análisis de los numerosos resultados publicados en los informes "IRIA: Tecnologías de la Información en las Administraciones Públicas" y "REINA: Las Tecnologías de la Información y las Comunicaciones en la Administración del Estado".

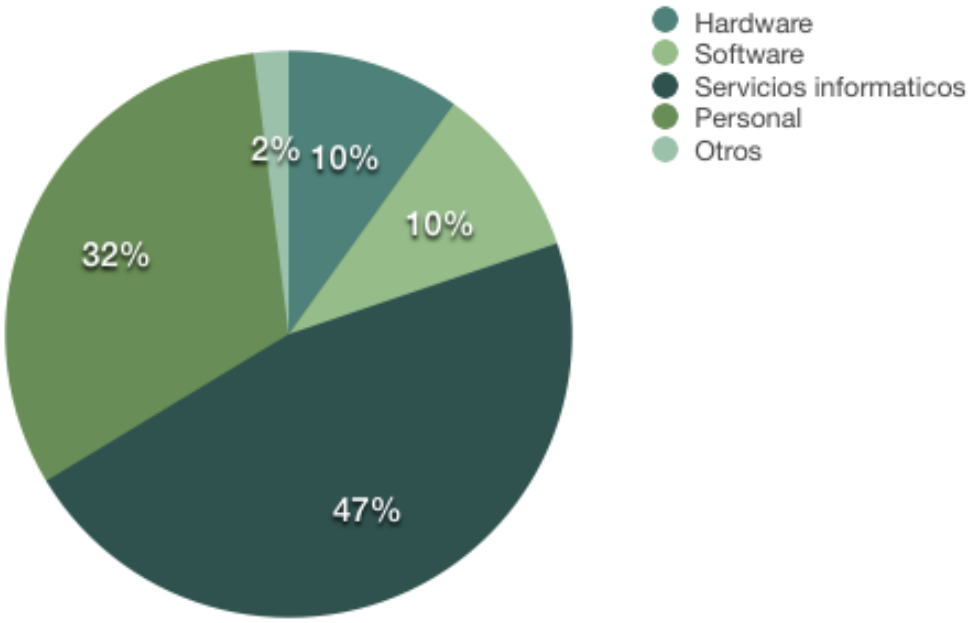
Según el informe REINA 2013, basado en datos del 2012, el gasto total en las Tecnologías de la Información y las comunicaciones se establece en 1.453 millones de euros un 13,6% menos que el año anterior. En la siguiente gráfica podemos observar la evolución del gasto en los últimos años:



Gráfica 6: Evolución del gasto en TIC de las Administraciones Públicas

Fuente: Informe REINAS 2013

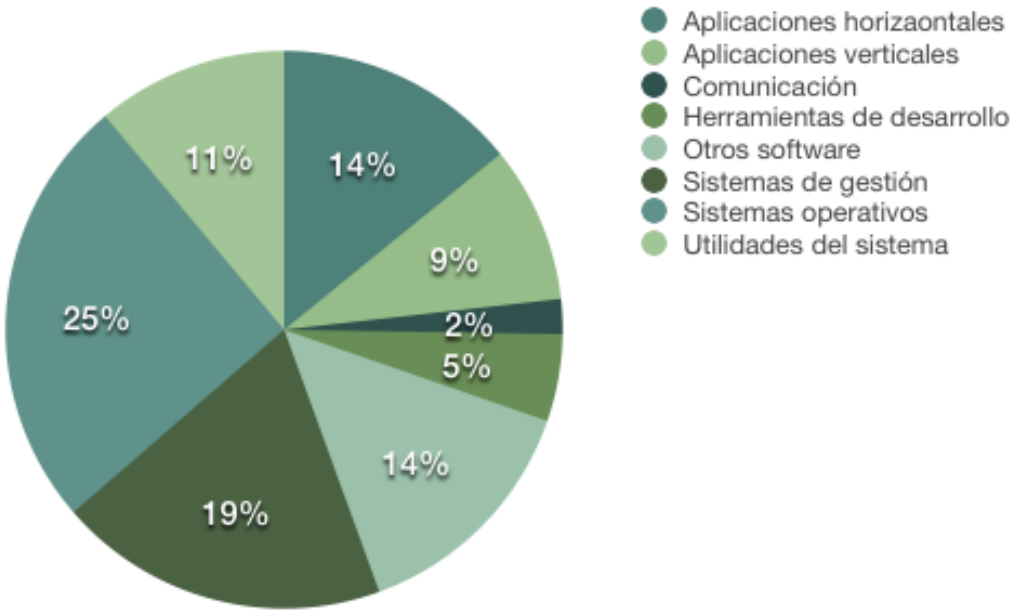
La cifra de 1.453 millones de euros se divide en, gastos informáticos la cual asciende a 1.077 millones de euros y el telecomunicaciones con un restante de 376 millones de euros. En el gráfico mostrado a continuación podemos observar la distribución del gasto en servicios informáticos por partidas. Un 47% corresponde a gastos en servicios informáticos (desarrollo y mantenimiento de aplicaciones, mantenimiento de hardware y software, gastos en consultoría, explotación, formación, ...), partida que desciende su participación dos puntos respecto a 2011. Los gastos de software, que incluyen la adquisición de software de sistemas y paquetes de aplicación, pero no los desarrollos a medida, incrementan un punto su participación hasta situarse en un 10% del gasto total.



Gráfica 7: Gasto por partidas de servicios informáticos.

Fuente: Informe REINAS 2013

A continuación nos ocuparemos de los gastos en software, los cuales ascienden a 107 millones de euros, los cuales se distribuyen según la siguiente gráfica.



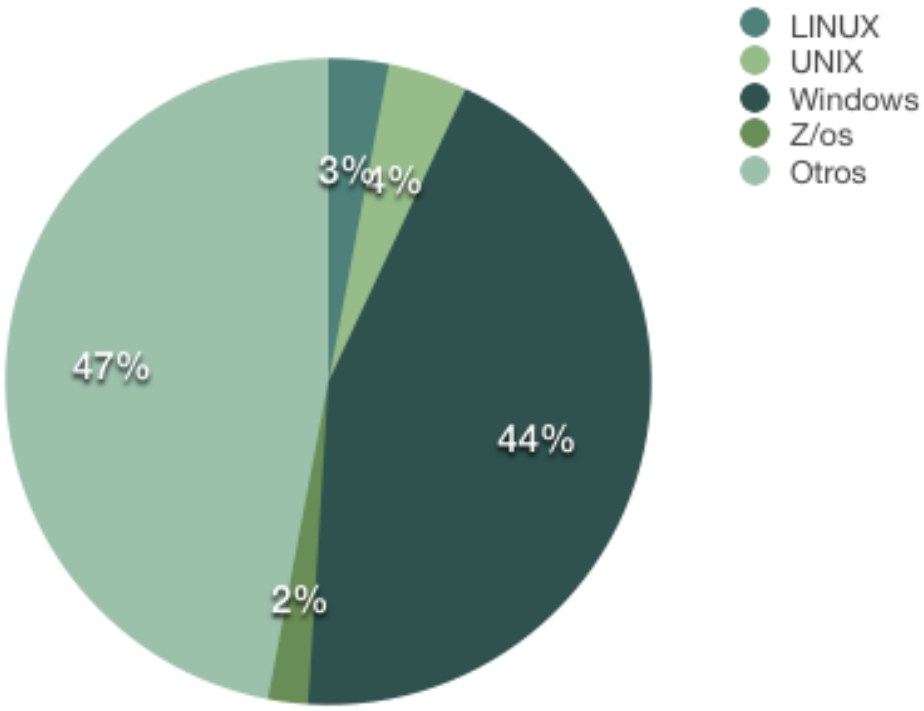
Gráfica 8: Gasto de Software por partidas

Fuente: Informe REINAS 2013

En esta edición el 25% de los gastos corresponde a sistemas operativos. Al igual que en ediciones anteriores hay que hacer notar que en muchos casos, especialmente en ordenadores personales, los gastos en sistemas operativos no se desglosan de los gastos de los equipos y, por tanto, no aparecen aquí reflejados. La adquisición de sistemas de gestión de la información supone el 19% del total de gastos en software mientras el 14% se destina a la adquisición de aplicaciones horizontales.

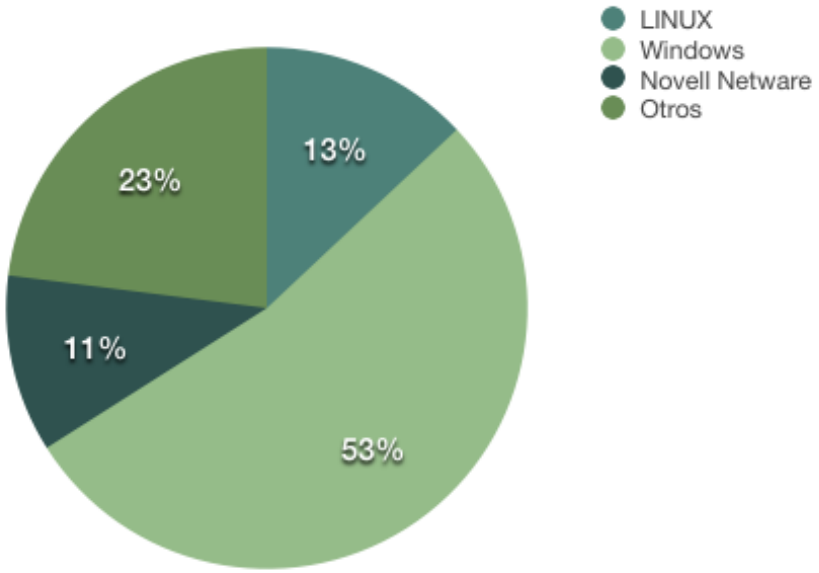
De los 472.989 equipos instalados a fecha enero del 2013 con 5 años de antigüedad, 45 corresponden a grandes equipos, de los cuales no se ha instalado ninguno en 2012, 1.058 corresponden a equipos medianos, de los cuales la mayoría (44%) utilizan Windows como sistema operativo, un 3% utiliza Linux y un 4% UNIX, como se observa en el gráfico 9.

Del total de equipos 21.178 son servidores, donde principalmente se sigue utilizando el sistema operativo Windows (53%), en cambio un 13% tiene instalado Linux como sistema operativo, gráfico 10.



Gráfica 9: Sistemas operativos en equipos medianos.

Fuente: Informe REINAS 2013.

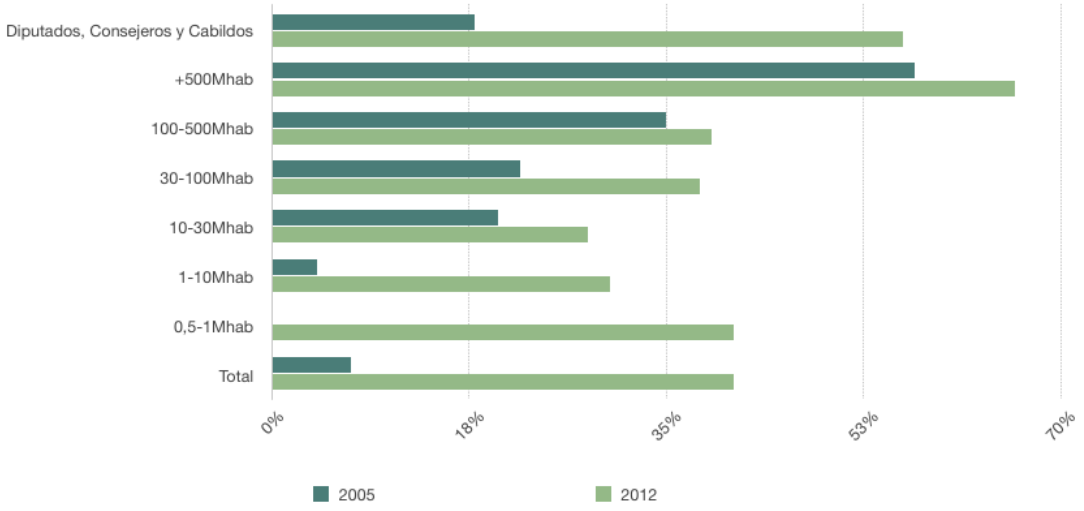


Gráfica 10: Sistemas operativos en servidores.

Fuente: informe REINAS 2013

A la vista de los datos anteriores, podríamos concluir que el gasto por parte de la Administración Pública en servicios informáticos se ha reducido considerablemente desde los últimos años, pero aun queda mucho camino por recorrer. Como hemos mencionado que a nivel local cada vez son más los Ayuntamientos que están emprendiendo inactivas de adopción de software de fuentes abiertas[12], centrándose especialmente en la migración de algunas de sus infraestructuras de comunicación, como los servidores web institucionales.

El Informe IRIA presenta un análisis de los indicadores más representativos de la situación y uso de los sistemas y tecnologías de la Información y Comunicaciones en la Administración Local, dedicando un apartado a analizar la presencia de políticas de adopción de software de fuentes abiertas a nivel municipal.



Gráfica 11: Comparativa uso de software libre 2005/2012.

Fuente informe IRA 2006 Y 2013

[12]El término técnico establecido por la Administración española a través de CENATIC como traducción de "open source"

A la luz de los datos publicados en los informes IRIA 2006 e IRIA 2012, sintetizados, vemos como los municipios de más de 500.000 habitantes han estado especialmente activos en cuanto al desarrollo de algún tipo de política de adopción de software de fuentes abiertas en los últimos años.

Dentro de este grupo de grandes municipios encontramos el mayor incremento ya que si en 2006 el 57% de municipios encuestados afirmaban desarrollar algún tipo de estrategia de adopción de tecnologías basadas en fuentes abiertas, un año después este porcentaje se incrementa hasta llegar al 66%.

El grupo de municipios de entre 30.000 y 100.000 habitantes aumenta en 16 puntos en el 2012 la cifra alcanzada en 2006, pasando de un 22% a un 38% de municipios que desarrollan este tipo de iniciativas.

El grupo de municipios de entre 1.000 y 500 habitantes a experimentado un gran cambio, pasando del 2006 al 2012 de un 0% a un 41%. Lo que manifiesta la implicación de los pequeños municipios en el uso del software libre.

2.8.4 Conclusiones datos cuantitativos

A la vista de los resultados anteriores, pueden extraerse dos conclusiones importantes respecto de la situación del software de fuentes abiertas en las Administraciones Públicas. La primera es que los sistemas operativos libres se están abriendo un camino en el campo de las principales máquinas que controlan los sistemas de información, donde son especialmente importantes parámetros como estabilidad, seguridad, fiabilidad, accesibilidad y facilidad de administración, los sistemas operativos libres son la opción preferencial. Por el contrario, en el apartado de ordenadores personales, parece que los usuarios siguen utilizando, mayoritariamente, sistemas operativos propietarios.

Por otro lado, no puede considerarse como mera coincidencia el hecho de que la mayor parte de las aplicaciones de software de fuentes abiertas, que se muestran como claras alternativas a otras soluciones propietarias acumulando una significativa cuota de penetración, sean precisamente aquellas capaces de funcionar con una compatibilidad, prácticamente total, sobre varios sistemas operativos (sean estos libres o no).

2.8.5 Análisis estratégico

A continuación expondremos un análisis estratégico basado en el método DAFO de la situación del software libre en las Administraciones Públicas españolas, en base a lo estudiado anteriormente.

El análisis DAFO es una herramienta que permite conformar un cuadro de la situación actual de la empresa u organización, permitiendo de esta manera obtener un diagnóstico preciso que permita en función de ello tomar decisiones acordes con los objetivos y políticas formulados. El término DAFO es una sigla conformada por las primeras letras de las palabras Debilidades, Amenazas, Oportunidades y Fortalezas, (en inglés SWOT: Strenghts,Weaknesses, Oportunities, Threats). De entre estas cuatro variables, tanto fortalezas como debilidades son internas de la organización, por lo que es posible actuar directamente sobre ellas. En cambio las oportunidades y las amenazas son externas, por lo que en general resulta muy difícil poder modificarlas.

2.8.5.1 Debilidades

- Falta de formación. Existe una importante carencia de formación en software libre entre los responsables de configuración, administración y gestión de sistemas de información en las Administraciones Públicas.

- Soporte y mantenimiento. Cuando la elección y selección de un programa de software libre no se ha realizado específicamente mediante contrato con una empresa que se responsabilice del mismo, no se puede garantizar su soporte y/o mantenibilidad, una vez instalado por los medios habituales.

- Falta de publicidad. Muchos grupos de desarrollo de software libre carecen de capacidad de marketing y publicidad de sus productos, hecho que no sucede en las grandes empresas de desarrollo de software propietario.

- Contratación orientada a productos. Los procedimientos de contratación de la Administración Pública están muy orientados a productos. Sin embargo, la mayoría de los modelos de negocio alrededor del software libre están muy orientados al servicio, por lo que suele ser más difícil cumplir con las

condiciones requeridas.

- Preinstalación de software. La compra de un nuevo equipo tiene asociada la adquisición forzosa de un determinado software (por ejemplo, un sistema operativo determinado) lo que supone que la sustitución de esos programas por software libre sea difícil de justificar, y de realizar.

- Necesidad de hardware específico. Los gastos derivados tanto de la adquisición de hardware compatible con software libre, como de la formación del personal sin experiencia en este tipo de soluciones, suelen ser un factor determinante a la hora de valorar el coste de las diferentes alternativas que pueden adoptarse.

2.8.5.2 Amenazas

- Patentes de software. Los intentos de legalizar la patentabilidad del software en la Unión Europea y otros países suponen la mayor amenaza para el software de fuentes abiertas y su modelo de desarrollo y distribución.
- Desconocimiento del modelo de licenciamiento. El desconocimiento general del tipo de licencias que utiliza el software libre implantado o utilizado puede provocar situaciones conflictivas, por ejemplo, al mezclar programas con licencias incompatibles, o en el momento de valorar bajo que licencia se va a liberar un determinado software.
- Normas discriminatorias. Hay casos en los que se adoptan estándares internacionales que discriminan al software libre. En este sentido, la labor realizada por el programa IDABC- es un buen paso para tratar de frenar esta amenaza potencial, que podría perjudicar la implantación del software libre en las Administraciones Públicas europeas.

2.8.5.3 Fortalezas

- Ahorro en despliegue. Ahorro considerable de costes de despliegue por el hecho de no tener que pagar un coste por licencia de las copias instaladas o número de usuarios que utilizan el programa.

- Estándares y protocolos abiertos. El software libre está normalmente basado en estándares y protocolos abiertos, definidos y accesibles públicamente.

- Auditoría de código fuente. En el caso de usar software de fuentes abiertas este requisito es más fácil de satisfacer mediante auditoría completa por una tercera parte independiente.

- Disponibilidad de una gran cantidad de programas. Las compañías comerciales necesitarían más de 12.000 millones de euros y más de 160.000 personas al año para producir el equivalente a lo que actualmente está disponible en software libre (Cenatic,2008).

- Reutilización. Una de las grandes ventajas del software libre es que, por regla general, a la hora de desarrollar una nueva solución se aprovecha gran parte del trabajo ya existente.

- Disponibilidad de conocimiento. Cualquier equipo de desarrollo o empresa puede tener fácilmente a su alcance los mínimos medios, plataformas de desarrollo y conocimientos necesarios para comenzar a desarrollar software libre avanzado.

- Rápida evolución. La capacidad técnica unida a una gran implicación personal de los desarrolladores, clave de los principales proyectos de software libre, garantiza un elevado nivel de actualización y mejora de funcionalidades.

- Posibilidades de adaptación. La capacidad de acceso al código fuente que ofrece el software libre permite disfrutar de un elevado nivel de adaptación del software a las necesidades concretas de las Administraciones Públicas.

2.8.5.4 Oportunidades

- Independencia de proveedor. Puesto que el software libre está basado en estándares y normativas abiertas, existe la opción de poder cambiar de proveedor de servicio en cualquier momento, escogiendo la solución más satisfactoria desde el punto de vista de coste y prestaciones ofrecidas.

- Interoperabilidad. El software libre, por utilizar estándares abiertos y ser público su código, ofrece la oportunidad de generar un marco de interoperabilidad entre las distintas Administraciones Públicas de un país, o de diferentes países de la Unión Europea, garantizando de esta forma el intercambio de información entre

las diferentes entidades públicas de cada estado.

- Transferencia tecnológica. El artículo 45 y 46 de la Ley 11/2007 establece la reutilización de aplicaciones y la transferencia tecnológica en el sector público. El uso de software libre en este sector facilita la aplicación de esta ley.

- Desarrollo industrial. El software libre supone una gran oportunidad para desarrollar en Europa una industria de software tecnológicamente innovadora y económicamente potente.

- Liberación de formatos y protocolos. Es posible que, mediante la liberación de especificaciones públicas o códigos de software libre puedan sustituirse los formatos exclusivos y cerrados que hasta ahora están bloqueando la adopción de software de fuentes abiertas en muchos entornos.

- Buena imagen entre los profesionales más jóvenes. El software libre goza de una imagen muy favorable, adquirida de forma espontánea, entre un gran número de ingenieros y profesionales del sector tecnológico, muy especialmente entre los profesionales más jóvenes, lo cual supone una oportunidad para su difusión.

Gráfica 12: Esquema análisis DAFO

Fuente: Cenatic. Elaboración Propia

3 El MODELO COCOMO

El Modelo Constructivo de Costes (Constructive Cost Model) fue desarrollado por B.W. Boehm a finales de los años 70, plasmado detalladamente en su libro "Software Engineering Economics".

El Modelo COCOMO es un modelo matemático basado en el número de líneas de código fuente (Boehm et al, 2000) Consiste en una jerarquía de modelos de estimación de costes que incluye los submodelos: básico, utilizado en aplicaciones de gestión cuya tolerancia a fallos es alta, intermedio, sistema operativo donde no se desea gran número de fallos, pero si ocurren pueden ser tolerados y detallado, utilizado en aplicaciones criticas donde obtener fallos puede derivar en graves consecuencias, como puede ser el software de los aviones.

- Submodelo básico: este modelo trata de estimar de una manera rápida y mas o menos burda, la mayoría de proyectos pequeños y medianos.
- Submodelo intermedio: En este submodelo se introducen 15 atributos del coste para tener en cuenta el entorno de trabajo. Estos atributos se utilizan para ajustar el coste nominal del proyecto a entorno real, incrementando la precisión.

  Los atributos del coste tratan de capturar el impacto del entorno del proyecto en el coste de desarrollo. De un análisis estadístico de más de 100 factores de influencia en el coste. Boehm retuvo 15 de ellos, estos se agrupan en cuatro categorías:

  - Atributos del producto: garantía, tamaño de la base de datos y complejidad del producto.
  - Atributos del ordenador: restricción de tiempo de ejecución, restricciones del almacenamiento principal, volatilidad de la máquina y tiempo de respuesta del ordenador.
  - Atributos del personal: capacidad del analista, experiencia en la aplicación, capacidad del programador, experiencia de la máquina virtual y experiencia del lenguaje de programación.
  - Atributos del proyecto: prácticas de programación modernas, utilización de herramientas software y plan de desarrollo requerido.

Cada atributo se cuantifica para un entorno de proyecto. La escala empleada es: muy bajo – bajo – nominal – alto – muy alto – extremadamente alto. Estos 15 atributos multiplican al valor nominal del proyecto adaptándolo así a su entorno.

- Submodelo detallado: este modelo puede procesar todas las características del proyecto para construir una estimación.

  Se introducen dos características principales:

  - Multiplicadores del esfuerzo sensitivos a las fases del proyecto. Por lo general algunas fases se ven más afectadas que otras por los atributos. El submodelo detallado proporciona un conjunto de multiplicadores de esfuerzo para cada atributo. Esto ayuda a determinar la asignación del personal para cada fase del proyecto.

  - Jerarquía del producto a tres niveles. Se defiende tres niveles de producto. Estos son módulo, subsistema y sistema.

A la vez, cada submodelo se divide en modos que representan el tipo de proyecto: orgánico, semiencajado y empotrado:

- Modo orgánico: un pequeño grupo de programadores experimentados desarrollan software en un entorno familiar. El tamaño del software varía desde unos pocos miles de líneas (tamaño pequeño) a unas decenas de miles (medio).

- Modo semiencajado: corresponde a un esquema intermedio entre el orgánico y el rígido; el grupo de desarrollo puede incluir una mezcla de personas experimentadas y no experimentadas.

- Modo empotrado: el proyecto tiene fuertes restricciones, que pueden estar relacionadas con la funcionalidad y/o pueden ser técnicas. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

Atributos	Valor					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extra alto
Atributos del software						
Fiabilidad	0,75	0,88	1	1,15	1,40	
Tamaño de base de datos		0,94	1	1,08	1,16	
Complejidad	0,7	,85	1	1,15	1,30	1,65
Atributos del Hardware						
Restricciones de tiempo de ejecución			1	1,11	1,30	1,66
Restricciones de memoria virtual			1	1,06	1,21	1,56
volatilidad de la máquina virtual		0,87	1	1,15	1,30	
Tiempo de respuesta		0,87	1	1,07	1,15	
Atributos de personal						
Capacidad de análisis	1,46	1,19	1	0,86	0,71	
Experiencia en la aplicación	1,29	1,13	1	0,91	0,82	
Calidad de los programas	1,42	1,17	1	0,86	0,70	
Experiencia en la máquina virtual	1,21	1,10	1	0,90		
Experiencia en el lenguaje	1,14	1,07	1	0,95		
Atributos del proyecto						
Técnicas actualizadas de programación	1,24	1,10	1	0,91	0,82	
Utilización de herramientas de software	1,24	1,10	1	0,91	0,83	
Restricciones de tiempo de desarrollo	1,22	1,08	1	1,04	1,10	

Tabla 4: Atributos modelo COCOMO.

Fuente: Boehm (1981)

3.1 COCOMO II

Los objetivos principales que se tuvieron en cuenta para construir el modelo COCOMO II fueron:

- Desarrollar un modelo de estimación de coste y cronograma de proyectos de software que se adaptara tanto a las prácticas de desarrollo de la década del 90 como a las futuras.
- Construir una base de datos de proyectos de software que permitiera la calibración continua del modelo, y así incrementar la precisión en la estimación.
- Implementar una herramienta de software que soportara el modelo.
- Proveer un marco analítico cuantitativo y un conjunto de herramientas y técnicas que evaluaran el impacto de las mejoras tecnológicas de software sobre los costes y tiempos en las diferentes etapas del ciclo de vida de desarrollo.

COCOMO II está compuesto por tres modelos denominados: Composición de Aplicación, Diseño Temprano y Post-Arquitectura.

- El modelo Composición de Aplicación se emplea en desarrollos de software durante la etapa de prototipación.
- El modelo Diseño Temprano se utiliza en las primeras etapas del desarrollo en las cuales se evalúan las alternativas de hardware y software de un proyecto. En estas etapas se tiene poca información, lo que concuerda con el uso de Puntos Función, para estimar tamaño y el uso de un número reducido de factores de costo.
- El modelo Post-Arquitectura se aplica en la etapa de desarrollo propiamente dicho, después que se define la arquitectura del sistema, y en la etapa de mantenimiento. Este modelo utiliza:
- Puntos Función y/o Líneas de Código Fuente para estimar el tamaño, con modificadores que contemplan el uso, con y sin traducción automática, y el "desperdicio" (breakage).
- Un conjunto de 17 atributos, denominados factores de costo que permiten considerar características del proyecto referentes al personal, plataforma de desarrollo, etc., que tienen injerencia en los costos.

- Cinco factores que determinan un exponente, que incorpora al modelo el concepto de deseconomía y economía de escala Estos factores reemplazan los modos Orgánico, Semiencajado y Empotrado del modelo COCOMO.

3.2 CONCLUSIONES

Durante la última década, la evolución de las tecnologías de desarrollo de software impulsó un nuevo enfoque en la estimación de costos, que considerara conceptos tales como orientación a objetos, reingeniería, reusabilidad, utilización de paquetes comerciales, composición de aplicaciones. Además, surgió la necesidad de que estos nuevos modelos se adaptaran a la granularidad de la información disponible en las diferentes etapas de desarrollo.

La familia de modelos de COCOMO II, constituida por los modelos *Composición de Aplicación, Diseño Temprano y Post-Arquitectura*, conforma estas premisas ya que son parte de sus objetivos principales.

COCOMO II, al igual que el modelo original preserva su estado de dominio público en relación a los algoritmos, la herramienta de software, estructuras de datos, relaciones e interfaces.

Para contar con información fidedigna y favorecer el continuo refinamiento y calibración del modelo, la USC implementó un *Programa de Recolección de Datos.*

Otra de las ventajas de este modelo es que puede ser adaptado (calibrado) a un organismo en particular, si se cuenta con la experiencia de un número importante de proyectos ya culminados que puedan aportar los datos necesarios para la recalibración

Sin lugar a dudas, en la actualidad siguen existiendo inconvenientes y limitaciones para las estimaciones, pero más allá de esto COCOMO II ha recorrido un importante camino, logrando la madurez necesaria del modelo para conseguir estimaciones de gran precisión.

4 MODELO CMMI

CMMI son las siglas de *Capability Maturity Model Integration*.

Es un modelo desarrollado por el *Software Engineering Institute* para la mejora de los procesos de las empresas de software que califica las compañías según su nivel de madurez.

- El CMMI esta formado por tres "constelaciones", es decir, tres alternativas diferentes:

- CMMI for Acquisition, empleado en la mejora de los procesos de contratación externa

- CMMI for Services, utilizado para el establecimiento y la entrega de servicios

- CMMI for Development empleada en la mejora del desarrollo de los productos, software.

El modelo CMMI establece un conjunto de prácticas clave, agrupados en áreas clave de proceso "key Process Area ", para cada una de estas áreas se define un conjunto de buenas prácticas, estas deberán de ser:

- Establecidas en un procedimiento documentado.
- La organización deberá de proveer de los medios y formación necesaria para su ejecución.
- Deben de ser institucionalizadas, es decir se deben de ejecutar de modo sistemático, universal y uniforme
- Medidas y controladas.
- Además de verificar su cumplimiento.

Por su parte, estas áreas de proceso se agrupan en cinco niveles de madurez:

- Nivel 0 – Inexistente: las organizaciones carecen completamente de cualquier proceso reconocible e incluso se desconoce la existencia de un problema a resolver.

- Nivel 1- Inicial: Las organizaciones no disponen de un ambiente estable para el desarrollo y mantenimiento de software. Aunque si utilicen tecnocas correctas de ingeniería, existe una falta de planificación. En este nivel el resultado de los proyectos es impredecible.

- Nivel 2 – Repetible: En este nivel las organizaciones disponen de unas prácticas institucionalizadas de gestión de proyectos, existen unas métricas básicas y un razonable seguimiento de la calidad.

- Nivel 3 – Definido: Además de una buena gestión de proyectos, en este nivel las organizaciones disponen de correctos procedimientos de coordinación entre grupos, formación del personal, técnicas de ingeniería más detalladas y un nivel más avanzado de métricas en los procesos.

- Nivel 4 – Gestionado: Se caracteriza porque las organizaciones disponen de un conjunto de métricas significativas de calidad y productividad, que se usan de modo sistemático para la toma de decisiones y la gestión de riesgos. El software resultante es de alta calidad.

- Nivel 5 – Optimizado: La organización está completamente volcada en la mejora continua de los procesos. Se hace uso intensivo de las métricas y se gestiona el proceso de innovación.

El modelo establece una medida del progreso, conforme al avance en niveles de madurez. Ya que cada nivel, a su vez cuenta con un número de áreas de proceso que deben lograrse. El alcanzar estas áreas o estadios se detecta mediante la satisfacción o insatisfacción de varias metas claras y cuantificables. Como sabemos, a excepción del primer nivel, cada uno de los restantes Niveles de Madurez está compuesto por un cierto número de Áreas Claves de Proceso (Key Process Area (KPA)). Cada KPA identifica un conjunto de actividades y prácticas interrelacionadas, las cuales, cuando son realizadas en forma colectiva permiten alcanzar las metas fundamentales del proceso. Las KPAs pueden clasificarse en 3 tipos de proceso:

- Gestión.

- Organizacional.

- Ingeniería.

Las prácticas que deben ser realizadas por cada Área Clave de Proceso están organizadas en 5 Características Comunes, las cuales constituyen propiedades que indican si la implementación y la institucionalización de un proceso clave es efectivo, repetible y duradero. Estas 5 características son:

1. Compromiso de la realización.
2. La capacidad de realización.
3. Las actividades realizadas.
4. Las mediciones y el análisis.
5. La verificación de la implementación.

Las organizaciones que utilizan el modelo, para mejorar sus procesos disponen de una guía útil para orientar sus esfuerzos.

4.1 CMMI for Development

El modelo tiene 4 áreas de conocimiento o disciplinas que incluyen: Ingeniería de Software (SW), Ingeniería de Sistemas (SE) Desarrollo Integrado de Productos y Procesos (IPPD) y Acuerdos con Proveedores (SS).

- Ingeniería de Software: Cubre el desarrollo de software y su mantenimiento,
- Ingeniería de Sistemas: Abarca el desarrollo total del sistema que puede o no incluir el desarrollo de software.
- Desarrollo integrado de Productos y Procesos: Contempla un enfoque sistemático para la colaboración de los involucrados relevantes a través de la vida del producto.
- Acuerdo con Proveedores: En proyectos complejos se requiere de la incorporación de proveedores para ejecutar funciones o añadir modificaciones a productos.

5 VALOR DEL SOFTWARE LIBRE EN LA ECONOMIA

Establecida la definición de software libre, a continuación estudiaremos el ahorro derivado del software libre, pero cuantificar los ahorros siempre ha sido algo complejo, más aun en el caso de las tecnologías de software libre, por lo que nos debemos plantear diferentes cuestiones:

- ¿Nos referimos a unos ahorros derivados de la reutilización del propio software, es decir a la reutilización de parte del código, que como sabemos es abierto, en otros software de distintas características?

- ¿Nos referimos a unos ahorros derivados del uso, por ejemplo Linux, LibreOffice o Drupal?

Ahorros por uso

En lo referente al uso, gran parte de las comunidades autónomas han declarado una importante reducción de costes conseguido gracias a la implantación del software libre. Obsérvese el caso de nuestra comunidad. El Principado de Asturias decidió en el 2004 la implantación de software libre en sus sistemas informativos, siguiendo con una progresiva implantación de Linux o la obligatoriedad de que todos los desarrollos realizados tengan como base un entorno basado en software libre. Se ha estimado que esta migración ha supuesto un ahorro en costes entorno al 35%. Otro ejemplo a destacar es la Xunta de Galicia, la cual ha declarado ahorros importantes por el uso de software libre. De acuerdo con los datos extraídos de el informe sobre el Estado del Software Libre en las Entidades de Galicia (2008), en la PYME gallega se ha duplicado el uso del software de fuentes abiertas, pasando de un 10% a un 23% en tan sólo un año. El incremento de utilización es todavía más notable en los centros de enseñanza, pasando de un 20% a un 65% de centros que usan software libre. Sin embargo es muy difícil diseñar un método que permita estimar un cifra de ahorros general por el uso de software libre, debido a la influencia de muchos factores, como son el tamaño de la organización, la situación heredada, las competencias profesionales o la necesidad de integración.

Para abordar esta cuestión CENATIC elaboró en 2010, con el apoyo de Emergya, Pablo Ruíz Múrquiz y Gartner Group una herramienta de estimación del "Coste Total de la Propiedad". Dicha herramienta proporciona una estimación comparativa (entre herramientas software libre y propietario) de los Costes y de los Ahorros a 5 años para los casos de migración y uso de software libre en escritorios, suites de productividad y sistemas operativos personales.

Ahorros por reutilización

En cuanto a los ahorros derivados de la reutilización es posible realizar una estimación a partir del análisis de datos públicos procedentes de diversas fuentes, como datos procedentes de investigaciones y estudios estadísticos realizados por la comunidad científica, diversas métricas de ingeniería de software disponibles, o la utilización de índices conocidos o estimables como el porcentaje de gasto en software en la Unión Europea o la distribución del mismo.

Este es el objetivo del presente estudio, el desarrollo de un modelo que permita, a partir de la utilización de las fuentes de información disponibles, la estimación de los ahorros derivados de del software libre, el cual proporciona a CENATIC y a las Administraciones Públicas una poderosa herramienta que, junto con otras metodologías, permiten minimizar los costes de desarrollo utilizando software previamente existente para las futuras licitaciones.

5.1 OTROS ENFOQUES DE MEDICIÓN

Estimar el ahorro o los beneficios económicos que la economía recibe de forma global del software libre es una tarea extremadamente complicada. Si partimos de la base de que los métodos de evaluación están basados en calcular las ventas, resulta que este punto de vista no tiene en cuenta un factor de vital importancia en el uso del software libre, como es, el uso de código abierto que no se vende, sino que se incorpora a través de trabajos internos, a través de servicios o está integrado en el sistema.

5.1.1 Enfoque basado en medir la contribución de las empresas

Este fue uno de los primeros enfoques, basado en la medición directa de la llamada "economía del software libre". Se contó con la información de las empresas que se autodefinen como proveedoras de servicios o de software de código abierto.

Un ejemplo de los problemas relacionados con este método se refleja en la discrepancia entre el valor calculado de la industria mundial de software libre, para el 2008 estimado en 8.000 millones de euros, (Pierre Audouin Consultants 2009).Y el estimado, por ejemplo, por HP, 2.500 millones de euros o IBM que asegura haber ingresado 4.500 millones de euros en conceptos relacionados con el software libre, (Eastern Management Group 2010). Por lo tanto este enfoque no sólo no contempla la contribución del software libre al mercado global, sino que ignora el software libre que está integrado en aparatos o sistemas, como por ejemplo el software libre que esta integrado en los teléfonos, automóviles, descodificadores, etc.

5.1.2 El enfoque del principio de sustitución

Un enfoque alternativo, llamado "principio de sustitución", se basa en intentar calcular el valor económico de toda la instalación de un software libre, sumando los recursos económicos que habría que invertir para sustituirlos por un sustituto "comercial/tradicional/privativo" de gama media.

Un ejemplo real de este enfoque fue un conjunto de estimaciones que elaboró el "Standish Group" (Cenatic 2013) y que en palabras del grupo contemplaron los siguientes pasos:

"Primero elaboramos una lista de los principales productos Software libre. A continuación nos dedicamos a buscar equivalentes comerciales. Calculamos el coste medio para la lista de los productos de código abierto por un lado y de los productos comerciales por otro y así para obtener un "coste comercial neto". Por último, multiplicamos el coste neto del producto comercial por nuestras estimaciones de instalaciones de los productos de código abierto".

Este enfoque es similar al utilizado por muchos organismos público y privados para estimar las pérdidas por la llamada "piratería". No obstante, hay dos motivos por los que este enfoque resulta muy poco fiable.

El primero, plantea el hecho de que no existe un sustituto perfecto del software. Afirmar que los precios son equivalentes cuando el producto no lo es, plantea un error que aumenta a medida que lo hace el grado de "no-sustituibilidad" de un producto (por ejemplo, LibreOffice puede ser percibido por ciertos usuarios como "incompatible" con otras suites de productos de oficina patentadas y en este caso sólo podrían considerarlo como un sustituto parcial).

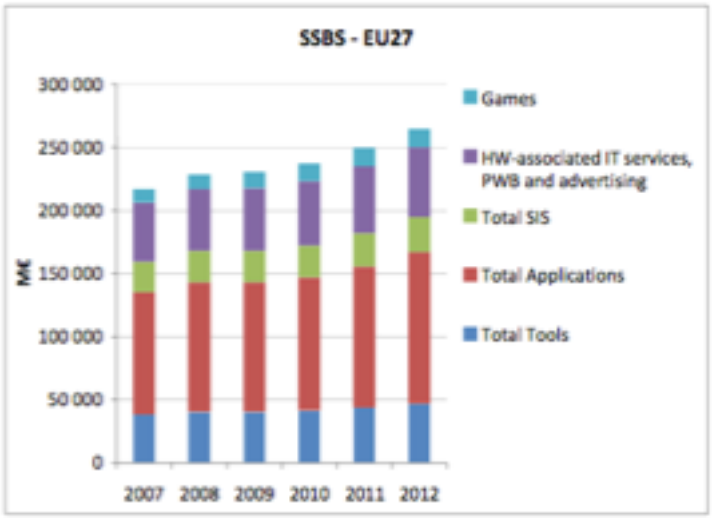
El segundo motivo está relacionado con el hecho de que puede que los usuarios no estén dispuestos a pagar por un producto alternativo al que ya tienen instalado, o quizás, no habrían adoptado el producto si hubieran tenido que pagar por él.

5.2 VALOR ECONÓMICO DE LA REUTILIZACIÓN DE SOFTWARE LIBRE

Otro método de estudio, podría basarse en el cálculo del ahorro, a través de datos relacionados con el grado de reutilización del software libre. Para ello, siguiendo la metodología de Daffara (2012) partiremos de varias encuestas sobre reutilización de código, combinadas con un conjunto de estimaciones macroeconómicas.

5.2.1 ESTIMACIONES MACROECONÓMICAS

Iniciaremos nuestro estudio, tomando como punto de partida todo el sector de las Tecnologías de la Información, el cual, fue valorado en aproximadamente en 624.000 millones de euros, durante el periodo del 2010. Basándonos en las estimaciones desarrollados por Pierre Audoin Consultants (2009) podemos determinar que aproximadamente el uso de las Tecnologías de la Información en Europa se establece en un 35% del uso mundial, ascendiendo a 224.000 millones de euros.



Gráfica 13: Mercado SSBS según los tipos de software para la Unión Europea.

Fuente: Pierre Audouin Consultants (2009)

En la gráfica anterior podemos observar el uso de los servicios de software en la Unión Europea, expresados según los diferentes tipos de software: juegos, servicios de Tecnología y la Información asociados, total de SIS, total de aplicaciones y total de instrumentos.

La estimación desarrollada por Pierre Audouin Consultants, ignora el equivalente económico de aquellos trabajadores, que aunque no estén contratados como personal del sector de las Tecnologías y la Información, si trabajan en este tipo de temas,

representando un gran porcentaje sustancial.

Basándonos en las estimaciones macroeconómicas de Gartner (2010), para tener en cuenta este valor "oculto", podemos estimar que este porcentaje asciende al 4% de la inversión directa o indirectamente imputable al sector de las Tecnologías de la Información.

Si nos referimos ahora a la Unión Europea, tan solo es necesario su PIB. Por lo que el equivalente económico de este porcentaje se estima en 150.000 millones de euros. Ascendiendo el valor económico del mercado de las Tecnologías y la Información a un total de 374.000 millones de euros.

Existen estudios paralelos, los cuales utilizan datos colectivos recogidos por asociaciones industriales del sector de las Tecnologías y la Información establecidas por todo el mundo. Posteriormente extrapolan dichos datos a la Unión Europea, obteniendo así un valor económico de aproximadamente 400.000 millones de euros.

Podemos apreciar que, este dato, no difiere mucho de nuestro valor económico, calculado a partir de los datos macroeconómicos de dicho sector y ajustado mediante los datos obtenidos por dos consultoras prestigiosas dedicadas a la investigación sobre la tecnología de la información.

Ahora bien, para obtener un valor económico verdaderamente objetivo, debemos de eliminar del valor total, aquel derivado de los esfuerzos de formación, atención al usuario, gastos financieros y otros servicios auxiliares, valorado este, por Pierre Audouin Consultants, en un 34,7% del total.

Finalmente, nuestro valor económico vinculado al software, se establece en 244.000 millones de euros. Cabe destacar que este valor económico es sustancialmente mayor que el del software "empaquetado" o lo que es lo mismo el software diseñado y desarrollado para venderse.

5.2.2 PORCENTAJE DE CÓDIGO ABIERTO PRESENTE

Llegados a este punto debemos de determinar, el porcentaje de código abierto correspondiente a este software, hecho que se aplica tanto a productos de software de desarrollo interno como a productos de software desarrollados y contratados externamente.

Varias fuentes revelan la existencia de cantidades sustanciales de código reutilizado:

- Black Duck: "Los resultados han revelado que un producto o aplicación contiene de media casi 700 MB de código, 22% de los cuales es código abierto. En otras palabras, casi el 80% del código que se está utilizando hoy en día es código abierto".
- La encuesta Koders descubrió en 2010 que el 44% de todo el código era código abierto.
- Sojer y Henkel revelaron una cuota de código reutilizado del 30% en los proyectos analizados.
- Veracode: "los estudios de muestreo siguen indicando que entre el 30 y el 70% del código presentado es de desarrollo interno y se identifica como propiedad de terceras partes, casi siempre bajo la forma de componentes de código abierto y bibliotecas comerciales".
- Gartner (2010) informó que entre los clientes investigados, el 26% del código que utilizaban era código abierto.

Basándonos en la información de Gartner y teniendo en cuenta que la utilización del código abierto aumenta con el tiempo, ya que se están utilizando cada vez mas proyectos abiertos, lo que implica un mayor uso de código reutilizado. Por lo tanto, para este estudio estimaremos que un 35% del código utilizado es código abierto.

Para tratar de proporcionar información sobre los ahorros que se produce tras la reutilización de software, utilizaremos como base, el modelos ya existente de estimación de costes, mencionado anteriormente, llamado COCOMOII, evidentemente, adaptado para los aspectos específicos de la reutilización de código abierto.

Como sabemos, este modelo está basado en un conjunto de estimaciones de coste distintas para cada sección independiente del proyecto. El modelo, tienen en cuenta el hecho de que reutilizar recursos externos, es decir, código abierto, introduce al proyecto, tanto ahorros como costes.

Por un lado, los ahorros son los derivados de poder reducir los esfuerzos de desarrollo, estos esfuerzos solo se reducen en parte, ya que como veremos a continuación es necesario emplear esfuerzos en acoplar el software de código abierto al software de código nuevo.

Por otra parte, los costes serían en concreto los derivados de:

- Costes variables, relacionados con el incremento del riesgo, debido a la falta de control de un recurso externo, como es en sí, la reutilización de código abierto.

  De aquí deriva, la importancia de participar en asociaciones y comunidades dedicadas a la gestión del código abierto, para así poder minimizar, en la manera que sea posible este tipo de riesgo.

- Costes derivados de la necesidad de adaptar y de crear el llamado "código pegamento" para poder integrar cada componente individual del software reutilizado con el resto del código nuevo.

Por tanto si utilizamos el modelo adaptado[13]-, llegaríamos a obtener una tabla resumida con los siguientes resultados:

Tamaño del proyecto (líneas de código)	% de OSS	Coste total (millones de euros)	Ahorros (%)	Duración (años)	Número medio de empleados
100000	0	1703	0	1,7	20,5
100000	50	975	43	1,3	15,4
100000	75	487	71	0,9	8,6
1000000	0	22000	0	3,3	141,7
1000000	50	12061	45	2,6	103,2
1000000	75	3012	86	2	32
10000000	0	295955	0	7,5	818
10000000	50	160596	46	5,9	631,2
10000000	75	80845	73	3,8	421

Tabla 5: Observaciones empíricas sobre el esfuerzo de integración del Software.

Fuente: Pierre Audouin Consultants (2009)

En la tabla podemos observar:

- Tamaño del proyecto, indicado a partir del número de líneas de código. El cual, para nuestro estudio, se ha establecido en tres cantidades fijas de líneas de código: 100.000, 1.000.000 y 10.000.000.
- Tanto por ciento de código abierto, establecido en el 0%, 50% y 75%

Obteniendo por lo tanto para un total de 100.000 líneas de código y un porcentaje de código abierto del 50%, un coste total de 975 millones de euros, con un porcentaje de ahorro del 43%. Resultado de haber trabajado durante aproximadamente un año y cuatro meses un total de16 trabajadores. podemos observar en la tabla, que a mayor numero de líneas de código, mayor será el coste y el ahorro derivados del uso de código abierto.

Anteriormente habíamos estimado la reutilización de código abierto en un 35%, por lo

[13] El modelo utiliza un porcentaje del 15% de código adaptado, con un indice de complejidad de 6 en comparación con los desarrollos de nuevo código. La volatilidad se calcula a partir de un esfuerzo adicional medio de entre 1,5 y 2,5 por persona empleada a tiempo completo por año.

tanto para determinar el coste total vinculado y el ahorro establecido. Debemos de sumar todos aquellos costes vinculados a la reutilización de código abierto, además del esfuerzo que requiere identificar y seleccionar los componentes del código abierto a emplear, este último coste es conocido como "coste de búsqueda". Estimando así para un 35% de código reutilizado, que el ahorro total equivale a un 31% del esfuerzo total de la inversión. Por lo que volviendo a los datos macroeconómicos obtenidos para la Unión Europea, podemos establecer un ahorro de aproximadamente 75.000 millones de euros año.

No obstante, este ahorro no es el único derivado de la reutilización de código abierto, sólo explica una pequeña parte del impacto.

Existen otros ahorros imputables a esta modalidad de reutilización, los cuales provocan una mejora del ratio de éxito[14] de un proyecto, como son:

- La reducción de los costes de mantenimiento
- La reducción del tiempo de lanzamiento
- La reducción de la reinversión en nuevos desarrollos, la cual es de especial importancia para la economía.

El ratio de éxito de un proyecto depende en gran parte del tamaño del proyecto y de los esfuerzos invertidos de el mismo, existiendo un gran número de proyectos que sufren retrasos o que se cancelan, esto da lugar a la aparición de una cierta "tasa de fracaso" en todos los proyectos, es decir, todo proyecto es susceptible de fracasar, bien porque su tamaño sea excesivo y de lugar a numerosos problemas o fallos, o bien porque la cantidad de recursos o personal empleado para el desarrollo sea insuficiente.

[14] Ratio de éxito, se denomina ratio de éxito al conjunto de parámetros tangibles y cualitativos tales como, el tiempo, el coste, el alcance, y valores mas cualitativos y en cierta medida intangibles como, aquellos parámetros relacionados con la satisfacción del cliente.

A continuación presentamos una serie de datos resumidos, los cuales reflejan el índice de éxito de los proyectos.

Tamaño	Personas	Tiempo	Índice de éxito
<750 millones de $	6	6	55%
De 750 000 a 1,5 millones	12	9	33%
De 1,5 a 3 millones	25	12	25%
De 3 a 6 millones	40	18	15%
De 6 a 10 millones	250+	24+	8%
> 10 millones	500+	36+	0%

Tabla 6: Datos de éxito de proyectos.

Fuente: Gartner Group (2010)

Según la encuesta realizada por Gartner Group entre sus clientes, podemos afirmar que cuanto mayor es el tamaño de un proyecto, y el numero de recursos asignados al mismo, más decrece el índice de éxito del mismo. A título orientativo, a continuación se presentan diversas citas:

- Jones revela que "el índice de cancelaciones de aplicaciones situadas en el rango 10.000 de la función es de alrededor del 31%. El coste medio de estos proyectos cancelados asciende a unos 35.000.000 de $".
- Grupo Standish, 2009: el 24% de los proyectos se cancelan antes de ponerse en marcha.
- Sauer & Cuthbertson, según una encuesta realizada por la Universidad de Oxford en 2003: el 10% de los proyectos son cancelados.
- Dynamic Markets Limited: más del 25% de todos los proyectos de software y servicios se cancelan antes de terminarse.
- Encuesta sobre el éxito de proyectos de TI de Dr. Dobbs: proyectos de Agile: el 60% tienen éxito, el 28% tienen problemas y el 12% fracasan; en el caso de los proyectos tradicionales, sin embargo: el 47% tienen éxito, el 36% tienen problemas y el 17% fracasan.

Como vemos existe gran cantidad de estudios sobre proyectos cancelados.

Volviendo nuevamente, al dato macroeconómico estimado para la Unión Europea y a nuestra aproximación la cual indicaba que el 35% de código empleado en los nuevos proyectos era código reutilizado. Podríamos indicar que la reutilización de software genera una reducción de la tasa de fracaso de aproximadamente un 2% sobre el esfuerzo total invertido, con un impacto económico de como mínimo 4.900 millones de euros.

Hay otro aspecto importante que guarda relación con los costes de mantenimiento, ya que el código que se reutiliza es sustancialmente mejor en términos de calidad, ya que se utilizan inconscientemente más recursos para su mejora.

Este hecho que fue observado por primera vez por Mohageghi, Conrad y Schwarz, "*el código que se reutiliza (como el código abierto) mejora más rápidamente y requiere sustancialmente menos esfuerzos que el código que no se comparte, lo que se traduce en que su esfuerzo de mantenimiento es considerablemente más bajo en medida."*

Si adoptamos el modelo de Jones y Bonsignour, CMMI[15], visto anteriormente, podremos determinar que el código tradicional posee un coste de aproximadamente 2.000€ por punto de función, mientras que el código compartido o desarrollado con las mejores prácticas tiene un coste menor, estimado en aproximadamente 1.200€ por punto de función.

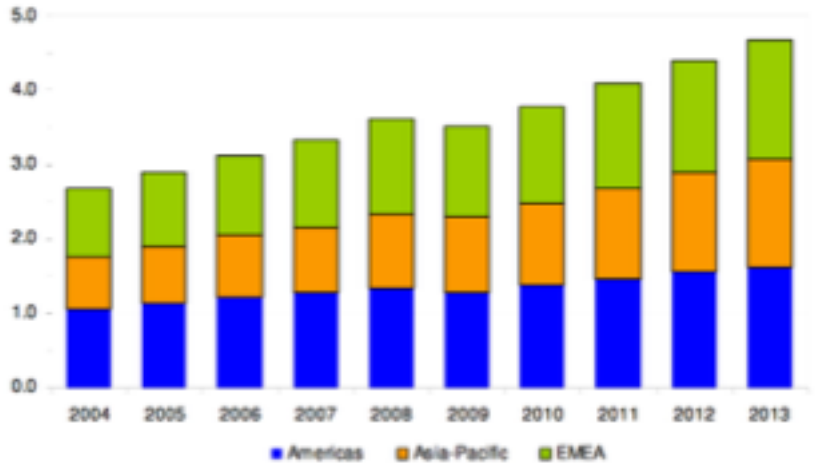
Por lo tanto, a partir de los valores obtenidos en el modelo de capacitación y madurez, podemos indicar, que estimando una reutilización de código abierto del 35% sobre el código total empleado. El uso de este código proporciona un ahorro de aproximadamente un 14%, en lo referente al concepto de mantenimiento y desarrollo. Lo que asciende para la Unión Europea en un ahorro adicional de 34.000 millones de euros.

Como habíamos mencionado anteriormente, un efecto de gran importancia para la economía es la reinversión de estos ahorros mencionados hasta el momento en el nuevos desarrollo de Tecnologías de la Información, hecho que se puede verificar al observar que el porcentaje de inversión en el desarrollo de las Tecnologías de la Información no disminuye, ni siquiera cuando aumenta el porcentaje de uso de software

[15] CMMI- Modelo de capacitación y madurez integrado

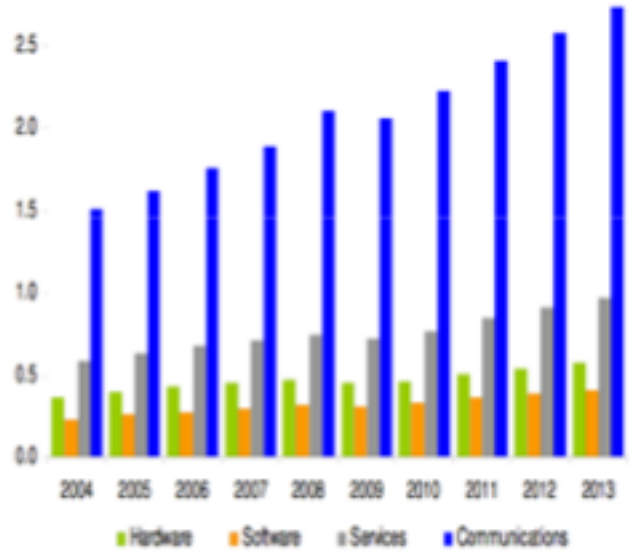
de código abierto

En las siguientes imágenes se muestra el aumento del gasto en el desarrollo de nuevas tecnologías.



Gráfica 14:Gasto estimado por región.

Fuente: WITSA Digital Planet (2010)



Gráfica 15: Aumento del gasto según los cuatro grandes grupos de la tecnología.

Fuente: WITSA Digital Planet (2010)

Como podemos observar en ambas imágenes se refleja el aumento del desarrollo de las tecnologías, aun aumentando el grado de utilización del código abierto. Lo cual es un indicativo de que los ahorros producidos por la reutilización del código libre, se

reinvierten en el desarrollo de nuevos proyectos.

El problema, es que estas reinversiones, desde el punto de vista macroeconómico, son casi neutras a corto plazo. Según datos de Brynjolfsson, podemos afirmar que a corto plazo la medida de la contribución de la "computerización" es aproximadamente igual a los costes de capital [16] - en bienes informáticos, mientras que a largo plazo es significativamente superior a los costes de capital en bienes informáticos.

Teniendo en cuenta que las empresas que utilizan mayor cantidad de software libre aprovechan más eficientemente esta medida de la contribución de la "computerización". Podemos afirmar que estas empresas deberían de gozar de un mayor grado de productividad o de eficiencia que sus iguales.

De hecho existe un estudio de Venice International University, que demuestra que al comparar los datos individuales de las compañías que tienen unos ingresos de menos de 500.000 euros y una variable definida por el tamaño de los clientes, como puede ser el número de empleados, podemos plantear la hipótesis de que existe una correlación entre el uso de código abierto y la capacidad para atraer clientes a una escala relativamente mayor que sus homólogas, estas sin uso de código abierto.

Gráfica 16: Compañías según ingresos y tamaño de sus clientes.

Fuente: Venice International University

[16] El coste de capital o coste de oportunidad de capital, indica aquella mínima tasa de rendimiento que permite a la empresa hacer frente al coste de los recursos financieros necesarios para acometer la inversión.

Como podemos observar en la imagen anterior, las compañías que utilizan software de código abierto y poseen más de 50 empleados son más eficientes para llegar a los clientes. O dicho de otro modo, al mismo nivel de ingresos las empresas que sólo utilizan software de código abierto parecen disfrutar de más ventajas para obtener encargos que las empresas con mas de 50 empleados, empresas medianas y grandes, que no utilizan software de código abierto.

Todo lo anterior, viene a justificar la hipótesis de que la reinversión de los ahorros tiene como efecto una mayor eficiencia. Si utilizamos los datos de Brynjolfsson, obtenidos a partir del modelo incremental anualizado de retorno de inversiones de capital, podemos, por tanto, estimar un resultado a largo plazo de mejora de la productividad y de la eficiencia como mínimo equivalente a 324.000 millones de euros en términos de valor añadido económico

6 SOFTWARE LIBRE COMO ESTRATEGIA EMPRESARIAL.

6.1 LINUX

El núcleo Linux es, sin duda, la aplicación estrella del software libre, hasta el punto de que, aún siendo una pequeña parte del sistema, ha dado nombre a todo él. Es más, se podría incluso afirmar que el propio software libre se confunde en multitud de ocasiones con Linux, lo que no deja de ser un gran desacierto, ya que existe software libre que corre sobre sistemas que no se basan en Linux. Por otro lado, también existen aplicaciones que funcionan en Linux y que no son software libre (Acrobat Reader, el lector de documentos PDF, también cuenta con su versión para Linux).

6.1.1 HISTORIA DE LINUX

La historia de Linux es una de las historias más conocidas dentro del mundo del software libre, seguramente porque se asemeja más a un cuento que a la historia de un programa de ordenador. En 1991, un estudiante finés de nombre Linus Torvalds [17]decidió que quería aprender a utilizar el modo protegido 386 en una máquina que su limitado bolsillo le había permitido adquirir.

Por entonces en los cursos de sistemas operativos universitarios existía un núcleo de sistema operativo gestado con fines académicos llamado Minix. A la cabeza del grupo de desarrollo de Minix se encontraba uno de los profesores de universidad más prestigiosos, Andrew Tanenbaum. Minix era un sistema limitado, pero bastante capaz y bien diseñado que contaba con una gran comunidad -académica e ingenieril a su alrededor.

Minix tenía una licencia de libre distribución y se podía utilizar sin problema para fines académicos, pero tenía el gran handicap de que personas ajenas a la Universidad de Amsterdam no podían integrar mejoras en él, sino que lo debían hacer de manera

[17] Mensaje original de Linus Torvalds al grupo de News disponible en Google Groups https://groups.google.com/forum/#!msg/comp.os.minix/dlNtH7RRrGA/SwRavCzVE7gJ

independiente, generalmente a través de parches. Esto suponía que en la práctica existiera una versión de Minix oficial que todo el mundo usaba y luego una larga serie de parches que había que aplicar posteriormente para obtener funcionalidades adicionales.

A mediados de 1991, Linus mandó un mensaje al grupo de noticias de Minix anunciando que iba a empezar desde cero un núcleo de sistema operativo basado en Minix, pero sin incluir código del mismo. Para entonces, Linus apuntó que el sistema que iba a crear no iba a tener las *barreras* con las que contaba Minix, por lo que sin saberlo, y probablemente sin quererlo, estaba dando el primer paso para quedarse con la comunidad que entonces se aglutinaba alrededor de Minix.

La versión 0.02 que data de octubre de 1991, aunque muy limitada, ya podía ejecutar terminales bash y el compilador GCC. En los meses siguientes el número de aportaciones externas fue creciendo hasta el punto de que ya en marzo de 1992 Linus publicaba la versión 0.95, que fue ampliamente reconocida como casi estable. El camino hacia la versión 1.0 fue, sin embargo, todavía largo, finalmente en marzo de 1994 Linux 1.0 vio la luz. Ya para entonces, Linux se publicaba bajo las condiciones de la licencia GPL, según el propio Torvalds una de las mejores decisiones que ha tomado, ya que ayudó sobremanera a la distribución y popularización de su núcleo.

6.1.2 EL MODO DE TRABAJO DE LINUX

La forma de trabajar de Torvalds no era muy común en aquellos tiempos. El desarrollo se basaba principalmente en una lista de correo. En la lista de correo no sólo se discutía, sino que también se desarrollaba. Y es que a Torvalds le gustaba sobremanera que toda la vida del proyecto se viera reflejada en la lista, por lo que pedía que los parches se enviaran a la lista. En contra de lo que uno pudiera esperar Linus prefería que se mandara el código en el cuerpo del mensaje para que él y los demás pudieran comentar el código. En cualquier caso, aún cuando muchos opinaran y enviaran correcciones o nuevas funcionalidades, lo que era indiscutible es que la última palabra, el que decidía lo que entraba en Linux, correspondía a Linus Torvalds. Hasta cierto punto, este modo de funcionamiento sigue vigente en 2014, en cambio junto a Linus se encuentra un pequeño grupo de colaboradores.

Otra de las ideas innovadoras de Torvalds fue el desarrollo paralelo de dos ramas del núcleo: la estable (cuyo segundo número de versión suele ser par, como por ejemplo 2.4.18) y la inestable (impar, como 2.5.12). Como no podía ser de otra forma, es Torvalds el que decide qué entra en uno y en otro sitio. En cualquier caso, Linux no tiene una planificación de entregas con fechas fijas, La decisión de si está listo o no corresponde, como no podría ser de otra forma, únicamente a Linus.

El método de desarrollo utilizado en Linux resulta, tal como se ha demostrado, muy eficaz en cuanto a resultados: Linux goza de una gran estabilidad y existe generalmente un intervalo ínfimo de corrección de errores (a veces del orden de los minutos), ya que cuenta con miles de desarrolladores. En esta situación, cuando existe un error, la probabilidad de que uno de ellos lo encuentre es muy alta y, en caso de que no lo pueda resolver el descubridor, alguno ya dará con la solución de manera rápida. En definitiva, podemos ver cómo Linux cuenta con miles de personas dedicadas a su desarrollo, lo que indudablemente hace que su éxito no sea nada sorprendente.

Se debe mencionar que esta forma de trabajar es, por contra, muy cara en cuanto a recursos se refiere. No es inusual que existan muchas propuestas mutuamente excluyentes para una nueva funcionalidad o que se reciban una docena de parches para el mismo error. En la gran mayoría de los casos, solamente una de ellas será incluida en el núcleo finalmente, por lo que se puede considerar que el resto del tiempo y esfuerzo dedicado por los desarrolladores ha sido en balde. El modelo de desarrollo de Linux es,

por tanto, un modelo que funciona muy bien en Linux, pero que ciertamente no todos los proyectos se pueden permitir.

6.1.3 ESTADO ACTUAL DE LINUX

Linux se encuentra en servidores y es aquí donde parece que está su mejor opción. Uno de los problemas de Linux se presenta en su escritorio, es necesario convertirlo en un producto más usable. Desde la comunidad de Software libre se entiende que algo no funciona en Linux, mientras que el Desktop Mac OS X ha crecido en muy poco tiempo del 3% al 7%, las distribuciones del aun siendo gratuitas no han pasado del 1%. Intentar revisar los errores y buscar soluciones a los problemas, todo ello para intentar ser relevante, y así poder disponer de unos sistemas más libres y abiertos, tanto de uso de internet como de gestión de la información mediante los ordenadores, es uno de los pasos a segur en el desarrollo del Linux. En cambio Linux se está utilizando mucho en otros tipos de dispositivos. Los televisores Samsung y Lg llevan Linux en ellos, algunos de Panasonic utilizan Unix Free BSD. Algunas programas de software libre como todos conocemos: Mozilla Firefox, Google Chrome y en menor medida Libre Office, han sido exitosos y debemos aprender todos de estos.

Uno de los principales problemas que se presentan en la incapacidad de Canonical, Red Hat y Novell[18]- de aumentar la cuota de mercado del escritorio, y ofrecer a Linux como una solución viable.

Paradójicamente, parece ser que Google puede ser una de las empresas que puede dar a Linux el reconocimiento que se merece. El mejor ejemplo lo tenemos con Android, Google en la siguiente tabla podemos observar que la mayor parte de los móviles utilizan el sistema operativo Android

[18] Canonical, es una empresa de software de ordenadores con base en Reino Unido destinada a la venta se software comercial y servicios relacionados. Red Hat, es la compañía responsable de la creación y mantenimiento de una distribución del sistema operativo GNU/Linux y Novell es una compañía de origen estadounidense dedicada al software, específicamente en el área de sistemas operativos de redes, como Novell Netware y Linux, es la empresa dueña de los derechos de la distribución SuSE Linux

Sistema Operativo	Participación en el mercado 2013
Android	81%
iOs	12,0%
Windows Phone	4%
BlackBerry	2%
Otros	6%

Tabla 7: Uso de los distintos sistemas operativos disponibles en el mercado.

Fuente IDC, elaboración propia.

Tenemos que tener en cuenta que ordenadores personales hay unos 1000 millones, pero como dijo Eric Smidth[19] *"móviles hay sobre tres veces más*". Es por ello que este mercado es muy importante, y es donde se le ha ganado la batalla claramente a Microsoft. Google ha adaptado Android de forma que funciona bien "Out of the box" y es usable, ha creado sus propias aplicaciones y ecosistema el cuál ha sido todo un éxito. La gente utiliza las aplicaciones y no se tiene que preocupar de nada porque todo va correctamente, ni siquiera saben que están usando un sistema operativo como es Linux. Aplicaciones como: Google Maps, Gmail, Google Earth, están muy bien desarrolladas, incluso en temas de usabilidad. Actualmente están vendiendo sobre 800.000 móviles con Android cada día.

Google es para mi la única empresa que tiene el poder de hacer que la cosa funcione bien y sea usable, además del poder de llevar a Linux al gran público. El trabajo de Red Hat o Novell ha sido, es y será muy bueno, pero Google es mejor en cuando a usabilidad y tiene más capacidad para llegar al usuario final, como ya se ha demostrado en más de una ocasión.

[19] Marketing Land, junio 2014, Mobile has won Disponible en: http://marketingland.com/google-chairman-mobile-has-won-69340

6.2 ANDROID

Android es un sistema operativo y una plataforma de software basada en Linux para teléfonos móviles, este sistema operativo se comienza a usar también en tablets, netbooks e incluso reproductores de música. Android ha permitido programar un entorno de trabajo (famework) de java, aplicaciones sobre una máquina virtual Dalvik[20], pero lo que le diferencia de otros es que cualquier persona que sepa programar puede crear nuevas aplicaciones o incluso modificar el propio sistema operativo, dado que Android es un software de código libre.

6.2.1 HISTORIA DE ANDROID

Android fue desarrollado por Android Inc., empresa que en 2005 fue comprada por Google, aunque no fue hasta 2008 cuando se popularizo, gracias a la unión del proyecto Open Handset Alliance, un consorcio formado por 48 empresas de desarrollo de Hardware, software y telecomunicaciones, que decidieron promocionar el software libre.

El sistema operativo Android, ha evolucionado rápidamente, que sea un software libre a favorecido al desarrollo de varias versiones, a continuación vamos a citar las distintas versiones:

- Android Nexus Edition: se trata del Android tal y como Google lo presenta en sus dispositivos Nexus. Esta versión está muy ligada a las aplicaciones de Google y podemos encontrar como muchos de sus servicios se han introducido, integrados como funciones básicas del teléfono.

- AOSP: la segunda forma de Android más importante en Android Open Source Project (AOSP). Se trata del Android para Nexus pero totalmente libre, esta versión es seguramente la mas importante. Se trata del origen del éxito del propio sistema y la forma de Android en la que todos se basan para fabricar las suyas.

[20] Dalvik es una máquina virtual (DVM), diseñada por Dan Bornstein, que permite ejecutar aplicaciones programadas en Java

- Android Corporation Edition: esta es la versión creada por Samsung, HTC, Sony, LG y demás no solo es en apariencia, sino que hay gran cantidad de líneas de código insertadas en sus núcleos que aumentan bastante las posibilidades de Android.

- ROMs: estas podrían considerarse un caso intermedio entre AOSP y el Android de los fabricantes. Básicamente son modificaciones de AOSP pero en vez de ser creadas por multinacionales la han sido por grupos de desarrolladores cuya principal característica es que también liberan el código, con lo que se produce una retroalimentación con AOSP.

- Forks:son una de las múltiples ramas de Android y también las encontramos en dispositivos no móviles como por ejemplo OUYA[21]. Se trata de la creación de un nuevo proyecto basado en el mismo código fuente pero el cual tiene una dirección distinta al original.

- Android TV: esta forma de Android permite convertir tu televisor en un televisor inteligente con multitud de funciones, navegar vía Wifi, consultar tu email, disfrutar de las Redes Sociales, jugar con la máxima calidad, reproducir vídeo Full HD, conectar tu pendrive y disfrutar de tus vídeos y música favorita,

- Android in the car: esta es una de las nuevas aplicaciones de Android, de la cual aun no se tiene suficiente información pero está llamada marcar una época en el sistema operativo. Google formo una gran alianza con los fabricantes de coches para integrar Android.

- Android para Google Glass: Google Glass es un dispositivo de visualización cuyo objetivo sería mostrar a los usuarios de teléfono inteligentes toda su información sin utilizar las manos, por órdenes de voz.

- Android Wear: finalmente la última forma de Android, se trata de Android para relojes inteligentes, su finalidad es facilitar el día a día de sus usuarios.

[21] OUYA, es una videoconsola de código abierto que funciona con el sistema operativo Android, pensada tanto para juegos de teléfonos móviles y juegos de la propia consola.

6.3 CASO DE ESTUDIO: GOOGLE-ANDROID

Existen muchos modelos de negocio que se están explotando alrededor del software libre, si bien es cierto que estos modelos no son los más habituales en la industria del software, ya que no es viable la venta de licencias de usos de programas. Sin embargo, se pueden utilizar modelos basadas en el servicio a terceros, como es el caso de Google y es sistema operativo Android.

Como sabemos Google es una empresa dedicada a la publicidad relacionada principalmente con la búsqueda. Mientras que Android es un sistema operativo diseñado para su uso en teléfonos móviles, aunque actualmente comienza a desarrollarse en otros muchos campos, como pueden ser: Tablets, e-book, o incluso nuevos desarrollos como Android Tv o Android Wear

Podríamos entender que no existe relación alguna, pero sin embargo si nos hacemos la siguiente pregunta: ¿Dónde comenzamos a hacer, cada uno de nosotros, la mayor parte de nuestras búsquedas? Enseguida obtendremos de respuesta. “En nuestros móviles”. Por lo tanto obviamente existe una gran relación entre Google como empresa de publicidad y Android como sistema operativo en los dispositivos móviles.

6.3.1 CINCO FUERZAS DE PORTER

Ahora bien, para trabajar más detalladamente este caso de estudio, recurriremos al modelo de las cinco fuerzas de Porter, desarrollado en 1979 por Michael E.Porter. Este modelo se utiliza para identificar y analizar las fuerzas que afectan un sector de la industria. Según el modelo, las empresas deben evaluar las oportunidades y amenazas que plantean los competidores potenciales, la disponibilidad de los productos sustitutos en el mercado y el poder de negociación conferida a sus clientes.

Para iniciar el estudio debemos de entender el cambio que esta dando el mundo de las telecomunicaciones, sabemos que cada vez es mayor el número de usuarios de dispositivos móviles, punto esencial en el cual radica el interés de Google en Android, trata de no perder su posición en el mercado, lo que podría ocurrir si se quedará anclado en el mundo de los ordenadores. Como veremos a continuación, con el modelo de las cinco fuerzas de Porter, Google ha desbancado a muchos de sus competidores e incluso, se puede decir que por el momento ha cerrado las barreras a nuevos, mediante su modelo de negocio basado en el sistema operativo Android.

6.3.1.1 Poder de negociación con los clientes

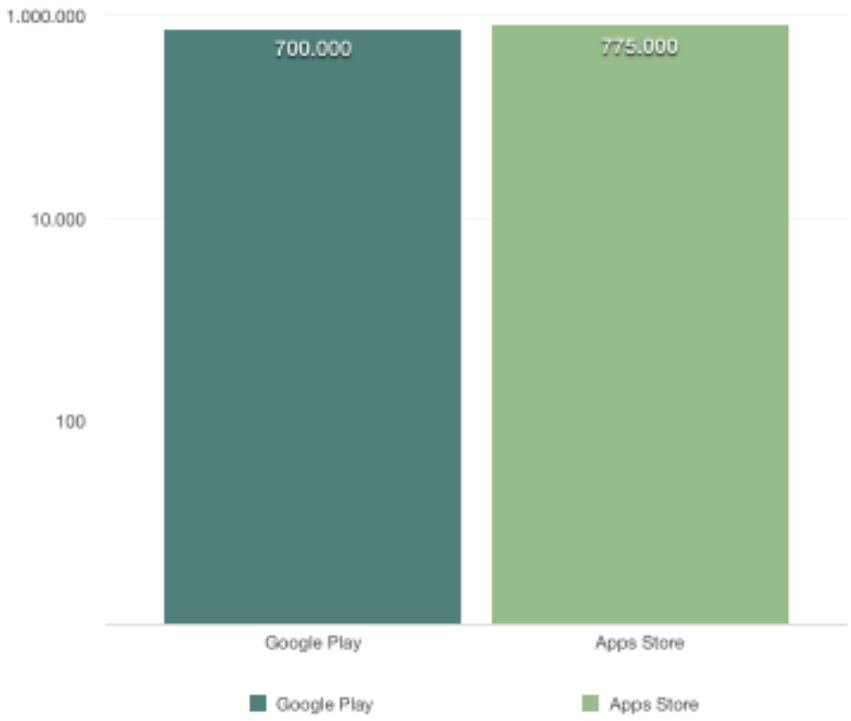
Para analizar los clientes de Google debemos de dividirlos en dos grandes grupos:

- Por una parte aquellos que disponen de Google para realizar las búsquedas, los cuales estan dispersos prácticamente por todo el mundo. Según StatCounter (2013) Google ostenta más del 90% de la cuota de mercado como motor de búsqueda y su cuota aún es mayor en la búsqueda mediante dispositivos móviles (97%) ya que prácticamente viene instalado por defecto en la mayor parte de smartphone.

- Y por otro lado aquellos clientes que disponen de Google para emitir sus anuncios publicitarios, los anunciantes, de los cuales podríamos decir que poseen un nivel de negociación mayor debido a sus inversiones en publicidad, este poder se ve reducido debido a que ellos están buscando acceder a un medio que llegue a la mayor parte del mundo, y lo que por el momento sólo pueden conseguirlo a través de Google.

Uno de los factores por los que Google ha llegado a tantos usuarios, es la simplicidad del sistema de búsqueda, bien es cierto que Google se ha convertido en la pantalla principal de millones de navegadores, e incluso podemos decir que si hacemos "click" en nuestro navegador y no aparece la pantalla de Google, inmediatamente pensamos que no hay internet.

Ahora bien ¿Podrá Google acercarse tanto a sus "nuevos usuarios"? como hemos visto anteriormente la estrategia de Google para estar presente en el futuro mercado ha sido el lanzamiento de Android, pero es necesario tener en cuenta que Android es un software libre, implantado en gran parte de dispositivos móviles, a manos de empresas de gran importancia como puede ser Samsung. Es por lo que Google no sólo ha lanzado un software libre, si no que este está acompañado de una serie de aplicaciones, implantadas de serie, como son Google Maps, Gmail y Docs, además de Google Play, no es más que una plataforma de distribución de aplicaciones móviles, desarrollada y operada por Google , lo peculiar de esta plataforma es que además de ser Google uno de los principales desarrolladores de aplicaciones, permite que desarrolladores del exterior desarrollen sus propias aplicaciones y las expongan a los usuarios, bien de forma gratuita o bien bajo el pago de un cierto canon.



Gráfica 17: Número de aplicaciones Google Play & Apps Store.

Fuente: The Sociable y Statista

Desde mi punto de vista, esto alienta a las otras plataformas como son App Store de Apple o Samsung Apps de Samsung, a seguir invirtiendo esfuerzos para desarrollar una plataforma que se asemeje a Google Play en la diversidad de contenidos. Según un estudio realizado por The Sociable y Statista (2013) Google Play, a pesar de llevar menos tiempo disponible, posee unas 700.000 aplicaciones disponibles de las cuales aproximadamente el 65% son gratuitas, frente a su principal competidor Apps Store en cual dispone de 775.000 de las cuales tan solo el 23% son gratuitas. Sin lugar a dudas esto le confiere a Google, junto con Android, una posición privilegiada en el mercado de la búsqueda en online, los que indudablemente, lleva a una de nuestras principales conclusiones: la gran mayoría de los anunciantes pagan altos cánones por publicar sus anuncios en Google, ya que están simplemente a un "click" de multitud de potenciales clientes.

Finalmente podemos concluir que debido al número de clientes, pero principalmente a al dispersión de los mismos, el poder de negociación que los clientes establecen sobre Google es bajo.

6.3.1.2 Poder de negociación de los proveedores

Para poder describir este apartado con mayor claridad recurriremos a la ley de conservación de los beneficios atractivos enunciada por Clayton Chistensen:

> "Cuando la modularidad y la comodificación hacen que los beneficios atractivos desaparezcan de un eslabón en la cadena de valor, la conservación de la integración implica que surgirá ña oportunidad de obtener beneficios atractivos con productos privativos en un eslabón adyacente"
>
> Chistensen (2013)

En lugar de hablar de cadena de valor nos referimos a red de valor, en la que los distintos integrantes aportan valor creando o modificando un producto o servicio, hasta que este llega al cliente final. Christensen indica que si un nodo de la red de valor se moduliza, es decir se separan los elementos que lo componen, y alguno de esos elementos se comoditiza, los beneficios que se estaban obteniendo pasan a manos de otro integrante situado en los nodos adyacentes de la cadena.

Veamos como Google ha modularizado la oferta de dispositivos móviles y ha comoditizado el sistema operativo Android, lo que ha permitido la aparición de nuevos y más baratos dispositivos, debido a que los precios han bajado, la demanda ha aumentado, por lo que se estimula el crecimiento de terminales Android, en los que los accesos a los servidores de Google se incorporan de serie. Google establece así fuertes relaciones con los fabricantes de dispositivos.

Por lo tanto, Google ha desarrollado Android con el objetivo primero de modulizar los dispositivos móviles, estableciendo una separación entre la fabricación de los aparatos físicos y de sus sistemas operativos, y segundo, comoditizar los sistemas operativos, para poder capturar parte de los beneficios, que antes correspondían a Nokia[22]- al vender conjuntamente los dispositivos y el software.

En conclusión podemos afirmar que el poder de negociación con los proveedores es

[22] Nokia era la empresa dominante en la fabricación de dispositivos móviles, y ofrecía sus terminales con sus sistemas operativos propietarios, hasta la llegada de Android.

alto. Ya que la mayor parte de los dispositivos móviles del mercado operan con el sistema operativo Android (81%).

6.3.1.3 Entrada de nuevos productos

Sin duda alguna la principal competencia de Android es Mac iOS , pero no podemos establecer que existe una competencia en si, ya que el sistema Mac iOS , establecido en los iPhone se posiciona en otro segmento del mercado de mayor calidad, para el cual no ha sido diseñado el software base de Android, es decir el software distribuido por Google. Podríamos decir, que iPhone establece cierta competencia con aquellas empresas que han utilizado como base el sistema operativo Android pero le han conferido ciertas mejoras (Android Corporation Edition), como pueden ser Samsung, HTC, Sony, LG.

Entre otros competidores podemos destacara WebOS, nuevo sistema operativo de Hewlett Packard, Windows Phone 7 u Omnia, el problema de estos últimos es que las posibilidades de los dispositivos donde van integrados, no se acercan a las ofrecidas por los fabricantes de dispositivos móviles con software Android confieren, esto se debe, en parte, a la reducción de costes de licencias del sistema operativo.

Los fabricantes pueden incrementar mejoras en el dispositivo lo que dará lugar a sus ventajas competitivas. Es decir, la comoditización, elimina los costes asociados a las licencias del sistema operativo, reduce las barreras de entrada en el negocio de fabricación de los dispositivos, lo que ha permitido la entrada de nuevos fabricantes en el mercado, revirtiendo en una mayor y más variada oferta de dispositivos, a su vez este incremento de la oferta, estimula la demanda de los dispositivos, y por tanto de su uso.

Con lo que Google gana la batalla siendo el motor de búsqueda más utilizado en los distintos dispositivo móviles.

Aunque no es de extrañar que se desarrollen otros nuevos sistemas operativos, como es el caso de: Ubuntu Touch desarrollado por Canonical, Sailfish desarrollado por una empresa finlandesa Jolla Ltd, Tizen desarrollado por Samsung y Firefox desarrollado por Mozilla Corporation, pero actualmente simplemente parecen una “puerta abierta” en el desarrollo de software para móviles, ya que los principales recursos de estas empresas, como es el caso de Samsung se destinan a la creación de nuevos dispositivos cada vez más desarrollados, pero como base el sistema operativo Android.

Por lo tanto podemos concluir que actualmente la entrada de nuevos productos es baja, aun que si bien es cierto que existen nuevas versiones, que en un mundo como es el de la informática, nunca se sabe el impacto que pueden causar.

6.3.1.4 Amenaza de nuevos competidores

Este punto esta muy ligado al anterior, como hemos visto nuevos sistemas operativos podrían intentar desbancar a Android, pero si hablamos de nuevos competidores, es decir, ¿sería fácil introducir en el mercado un nuevo software libre, similar a Android, que generara una amenaza para Google? Para responder a esta pregunta debemos de volver a la base del negocio de Google. Como bien sabemos Google es una empresa dedicada a la publicidad y todo lo que hace Google debe entenderse en este contexto. Google desarrolla servicios como Google Maps, Gmail y Docs y los cede de forma gratuita, esto podría tener base en una creencia filosofía basada en que las aplicaciones deben de ser libres; o puede estar siguiendo una estrategia empresarial, ya que dar "regalos" gratis le confiere una gran ventaja frente a sus competidores, es decir, que Google regale aplicaciones, provoca que más gente las usemos, que si realmente hubiera que pagar por ellas, por lo tanto si Google ofrece un medio de publicidad que puede llegar a todos los rincones del mundo, obviamente podrá obtener mayores ingresos por publicidad.

Como hemos iniciado anteriormente, al regalar las aplicaciones como: Google Maps, Gmail y Docs, hace que aumente el número de usuarios, pero también evita que sus competidores le hagan frente, ya que para las compañías es muy difícil competir en esa área. El esfuerzo necesario sería demasiado elevado para que las empresas actuales pudieran hacerle frente, como hemos mencionado anteriormente, la ventana de Google es el enlace a internet, eso le confiere una credibilidad como gran empresa que es difícil de alcanzar.

6.3.1.5 Competidores del sector

Como hemos indicado en los puntos anteriores la estrategia empresarial que Google ha llevado a cabo con el uso del software libre, le ha llevado a reducir en gran parte el número de competidores. Como hemos estudiado Google se ha colocado en una posición de éxito para la nueva etapa de la publicidad relacionada con la búsqueda, que como hemos apuntado anteriormente tiende a crecer en el uso de los dispositivos

móviles.

Por lo que finalmente podemos concluir que Google está construyendo Android con dos objetivos:

- Interrumpir el crecimiento del dominio de los dispositivos móviles de Apple.
- Controlar la industria móvil y por lo tanto la publicidad de ello.

Según Mihail L. Sichitiu (2010) Android es actualmente el único sistema operativo de código abierto importante para teléfonos móviles y es proporcionado sin costo alguno para los fabricantes de teléfonos con un considerable apoyo de Google. Estas dos características han impulsado los teléfonos con Android en dos años aproximadamente sobre un mercado prácticamente inexistente.

A la vista de los resultados obtenidos anteriormente, ahora debemos de preguntarnos: ¿Ha sido Android, desarrollado por Google, con animo o sin animo de lucro?

Según Schmidt (2009) en una entrevista realizada por Erick Schonfeld de la empresa techcrunch a Eric Schmidt CEO (Chief Executive Officer/Gerente General de Google), proyecta a Android como un sistema operativo que genera suficiente dinero como para cubrir los costos de desarrollo, y con su crecimiento esperan que represente la mitad de los ingresos actuales de Google, rompiendo con los esquemas de muchos expertos que describían este sistema operativo como "una pérdida de tiempo".

Por otro lado de acuerdo a Stone (2010), en una entrevista realizada por el New York Times a Andy Rubin, Vicepresidente de Ingeniería de Google, enfoca el desarrollo de Android como una plataforma libre para dispositivos móviles, expone lo siguiente: "When they can't have something, people do care" (cuando la gente no tiene acceso a algo, le importa) refiriéndose a la principal característica del sistema operativo Android, el cual implica la libertad de poder interactuar y modificar el sistema operativo en todos sus niveles, siendo esta la gran diferencia ante sus competidores y su arma elemental para vencer el mercado de los dispositivos móviles inteligentes. Rubin opina que el hecho de que Android sea "*abierto*" implica un gran atractivo para sus consumidores ya que son ellos los que exigen tener mayor acceso a la plataforma de los dispositivos móviles.

Por lo tanto, lo concluido anteriormente, esta en consonancia con las opiniones de grandes expertos. Conocemos que Google, como motor de búsqueda por internet, ha conseguido posicionarse como en dispositivos móviles gracias a Android. Pero ha de quedar patente que aunque ha suministrado libremente el software Android, si ha obtenido grandes beneficios debidos a la venta de aplicaciones o servicios adicionales. Según Sichitiu (2010) Android es actualmente el único sistema operativo de código abierto importante para teléfonos celulares y es proporcionado sin costo alguno para los fabricantes de teléfonos con un considerable apoyo de Google.

Lo que obviamente genera una gran competitividad en el mercado, siendo esta competitividad una de las principales premisas del software libre.

Por lo que podemos concluir que el desarrollo de Android ha sido beneficioso para Google, ¿Pero ha sido menos beneficioso para Samsung? Sin entrar en gran detalle podemos apreciar que Samsung se ha convertido en el principal competidor de Apple, posee unos dispositivos de gran gama, basados en el sistema operativo Android. Por lo que sin lugar a duda, Android también ha beneficiado a otras grandes empresas tanto a Samsung, como a las propias empresas responsables de las imitaciones de estos dispositivos.

Por último nos podríamos preguntar ¿Por qué no desarrollar un fork de Android que compita con el propio Android? Recordemos que un fork es cuando se parte del sistema operativo madre pero se realizan cambios, lo que nos lleva a otro "nuevo" sistema operativo, por ejemplo Linux y Mac OS X son un fork de UNIX.

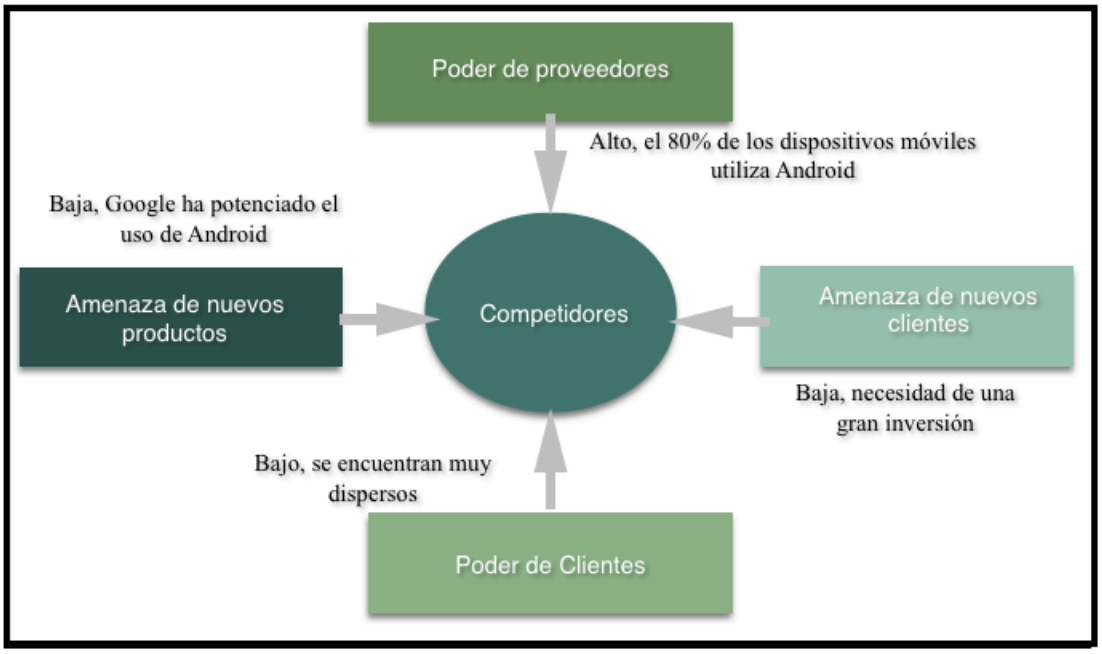
Para intentar responder a la pregunta anterior debemos de tener en cuentan distintos aspectos:

- El primero de ellos, es que actualmente no podemos considerar a Android como un simple sistema operativo, ya que consiste en toda una comunidad de trabajadores, unidos por Google Play, al cual un fork de Android no podría acceder, por lo que para igualar la oferta necesitaría una gran inversión de esfuerzos, lo que a corto plazo no parece posible.
- Otro punto importante sería el acceso a los servicios ofrecidos por Google, incluyendo Google Maps, y debemos de tener en cuenta que Google Maps encabeza la lista de las aplicaciones más utilizadas desde nuestros dispositivos

móviles (Keynote Systems, 2012).

- Finalmente debemos de considerar que Android cuenta con el respaldo de los fabricantes, ya que además de contar con una gran empresa a sus espaldas como es Google, Android tiene unos cimientos y unas bases que lo han llevado al éxito.

Por lo que podemos concluir que en un corto plazo de tiempo, será difícil que un fork de Android le pueda hacer frente, pero si bien es cierto que a largo plazo llegará, lo cual, no creo que pueda ser malo para Android, ni mucho menos para sus usuarios, porque la presión ejercitada por este nuevos sistema operativo daría lugar a una competencia, que indudablemente llevaría a Android a invertir en mejoras.



Gráfica 18: Esquema fuerzas de Porter.

Fuente: elaboración propia

7 CONCLUSIONES

La elección de este trabajo supuso un gran desafío en mi carrera, debido al desconocimiento inicial sobre el tema. Los inicios fueron especialmente difíciles, a causa de las distintas terminologías y la evolución del mismo durante los últimos años. Pero gracias a la importancia que, para mi, posee el software libre en la economía futura, el desarrollo del proyecto siguió adelante, completando mi formación.

El transcurso de este master, ha supuesto en mi formación un gran cambio, durante mi carrera, había aprendido a llevar a cabo el desarrollo de proyectos, tanto de pequeña como de gran envergadura. Pero el paso por el master y la realización de este proyecto, me han aportado, una visión económica y estratégica, es decir, la importancia de la realización previa de estudios a nivel comercial, estratégico o financiero.

En el presente proyecto se ha tratado de sintetizar aspectos importantes del Software libre, tanto su definición, como sus ventajas. Tratando puntos importantes como su existencia y el uso de el mismo en las Administraciones Públicas de nuestro país, así como los ahorros derivados de la implantación del mismo y la su utilización como base de una estrategia empresarial. Por lo que nos deja muchas reflexiones interesantes a su paso.

Debemos de dejar claro que cuando hablamos de software libre (free software /open source), estamos hablando de un software que permite su modificación y libre distribución. Pero este no tiene porque ser gratuito, aunque es habitual, que de existir un pago, este sea bajo. El software libre esta desarrollado generalmente por *comunidades* de jóvenes desarrolladores, los cuales utilizan como medio de comunicación internet.

De la definición de software libre, se deriva una de las principales conclusiones, la cual da respuesta a la pregunta: ¿Cómo existe algo que en un principio parece no responder a las leyes de la economía? Pues bien, después de estudiar las distintas motivaciones de los desarrolladores y los distintos métodos de financiación que hacen posible la existencia de dicho software, podemos concluir que, el software libre no se desarrolla de forma altruista, sino que responde a las necesidades de un gran numero de interesados *skateholders*. Estas necesidades son entre otras:

- Desde el punto de vista de los desarrolladores, necesidad de obtener conocimientos, ya que hemos observado que gran parte de los desarrolladores se encuentran en su periodo de formación. Pero también necesidad de darse a conocer en su ámbito de trabajo, ya que ser participe de un software libre, el cual se distribuye rápidamente, puede reportar al desarrollador un gran reconocimiento.
- Desde el punto de vista de las distintas instituciones, empresas o sociedades, las cuales aportan la financiación necesaria para la existencia del mismo, sus necesidades suelen cubrir aspectos como la necesidad de un software que permita el desarrollo de otro estudio, la necesidad de un software que implemente a un hardware o también la necesidad de ofrecer un servicio.

Por lo tanto, en mi opinión y en la de muchos autores, citados en dicho proyecto, el software libre, nace del deseo de crecer profesionalmente de sus desarrolladores y de las oportunidades de negocio que ofrece a las entidades financiadoras.

Otro punto a tratar en dicho proyecto, es el uso del software libre por parte de las Administraciones Públicas del país. Podemos concluir que del uso del software libre se deducen grandes ventajas significativas, destacando el ahorro de costes, la independencia del proveedor, la existencia a largo plazo e incluso la estimulación del mercado local, pero si bien es cierto, que para la obtención de dichos beneficios, es necesario previamente una planificación de su implantación.

Aunque el desconocimiento por parte de los altos cargos de las Administraciones y la falta de mecanismos de contratación de software libre dentro de las Administraciones Públicas genera grandes dificultades para su implementación, es cierto que, en nuestro país existe un gran número de ejemplos que ponen de manifiesto las ventajas del uso del mismo.

Además de la importancia que el software libre posee en las Administraciones Públicas, en este proyecto, se ofrece también la visión por parte de la empresa privada, para la cual, el uso del software libre supone también multitud de ventajas, como se observa en el caso de estudio realizado sobre la estrategia empresaria, llevada a cabo por Google, basada en el uso del software libre "Android". Punto en el cual podemos concluir:

- Que un software libre, generalmente es desarrollado para alcanzar un fin mayor, véase que Google se coloca en una posición privilegiada para afrontar la nueva etapa de las comunicaciones.
- Es necesario destacar que la creación de Android además de beneficiar a Google, ha beneficiado a muchas otras empresas, con lo que Google ha controlado el mercado de los dispositivos móviles, lo cual parecía ser su principal objetivo, ya que con ello consigue el control de la nueva era publicitaria.

El proyecto se completa con una valoración económica del software libre en la economía de la Unión Europea, de la cual, podemos concluir:

- Los datos recopilados hasta la fecha indican que el software libre posee un efecto positivo inmediato en la economía, gracias a las prácticas de reutilización del mismo así como a la reducción de esfuerzos en su desarrollo.
- Una estimación, basada en datos macroeconómicos y una serie de encuestas sobre la reutilización del software manifiestan un resultado de 114.000 millones de euros al año, como mínimo, gracias tanto al impacto de los ahorros directos como a la reducción de la tasa de fracaso de los proyectos.
- El efecto de reinvertir estos ahorros de forma interna, genera un efecto adicional en términos de productividad y de mejora de la eficiencia de al menos 342.000 millones de euros al año.

Como punto final a este trabajo, incluiré una pequeña opinión personal respecto al futuro de este tipo de programas, que sin restricciones de modificaciones o de distribución, e incluso por su calidad, esta en posición de establecer una nueva etapa tanto en el desarrollo del software, ya que hablamos de software desarrollado por multitud de desarrolladores trabajando en equipo pero, aportando cada uno de ellos, con la mayor de las libertades, sus necesidades, lo que confiere un software de gran calidad.

Junto a lo anterior, su función en la economía de nuestro país, con sus innumerables aportaciones, puede establecer importantes cambios en todos los niveles, desde la mejora de los negocios locales, hasta la reducción de gastos por las Administraciones Públicas. No obstante es necesario, indicar que para que esto suceda, el software libre debe de romper una de sus mayores barreras, tal como el desconocimiento, debido a la falta de información y ausencia de publicación de los distintos casos de éxito.

8 BIBLIOGRAFIA

- Affero general public license (2002) Disponible en: www.affero.org/oagpl.html. Revisada: 27.02.2014

- **ALIAL** (2010). Guía de buenas prácticas para la licitación de desarrollos libres por parte de administraciones locales. Proyecto Alial, desarrollado por Centro Nacional de Referencia de Aplicación de las TIC (CENATIC) & Federación Nacional de Empresas de Software Libre (ASOLIF). Disponible en: http://www.osimga.org/gl/documentos/d/2010_11_17_guia_alial_v1.pdf

- **AUMASSON**, A., **BONNEAU**, V., **LEIMBACH**, T. y **GÖDEL**, M. (2009) "Economic and Social Impact of Software and Software Based Services" Pierre Audouin Consultans agosto 2010. Disponible en: http://cordis.europa.eu/fp7/ict/ssai/docs/study-sw-report-final.pdf

- **BASILI**, V.R., **BOHEM**, B. (2001) "COST-based systems top 10 list" Computer, January 2001, pp. 135-137

- **BELLERFLAMME**, P. **LAMBERT**, T **SCHWIENVACHER**, A. (2012) "Crowdfunding: Tapping the right crowd" Journal of Business Venturing, Forthcoming, July 9, 2013

- **BOENHM**, B., **ABTS**, A.C., **BROWN**, W., **CHULANI**, S., **BRADFORD** K. C., **HOROWITZ**, E., **MADACHY**, R., **REIFER**, D.J., and **STEECE**, B., (2002) "Software Cost Estimation with COCOMO II". Englewood Cliffs, NJ:Prentice-Hall,

- **CARNEGIE MELLON** University Software Engineering Institute (2010). "CMMI for Development, Version 1.3". Disponible en: http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=9661

- **CARROLL** A.B. (1979) "A Three-Dimensional Conceptual Model Of Corporate Social Perfotmance" Academy of Management Review 4,4

- **CARROLL** A.B. (1991) "The Pyramid of Corporate Social Responsibility: Toward the Morl Management of organizational Stakeholders" Business Horizons, July-August 1991

- **CENATIC** (2008) "Software de fuentes abiertas para el desarrollo de la administración pública española, una visión genera" Disponible en: http://observatorio.cenatic.es/index.php?option=com_content&view=article&id=39:software-de-fuentes-abiertas-para-el-desarrollo-de-la-administracion-publica-espanola-una-vision-global-2008&catid=5:administraciones-publicas&Itemid=21

- **CENATIC** (2010) "Impacto de la reutilización del Software de fuentes abiertas en la economía" Disponible en: http://www.cenatic.es/publicaciones/onsfa?download=144%3Aimpacto-de-la-reutilizacion-del-software-de-fuentes-abiertas-en-la-economia.

- **CENATIC** (2011) "Uso del software libre en la empresa española 2010-2011". Disponible en: http://www.cenatic.es/publicaciones/onsfa?download=113%3Auso-del-software-libre-en-la-empresa-espanola-2010-2011.

- **CRUZ**. A. (2010) "Apuesto por Android como líder en aplicaciones móviles" Revista Científica Electrónica Ciencias Gerenciales / Scientific e-journal of Management Science Disponible en: http://www.revistanegotium.org.ve/pdf/19/art4.pdf

- **DAFFARA**, C. (2010) "The economic Value of Open Source Software" Submitted presentation, TransferSummit Oxford 2010

- **DAFFARA**, C. (2011) "Estimating saving from OSS code reuse, or where does the money come from". Disponible en: http://carlodaffara.conecta.it/estimating-savings-from-oss-code- reuse-or-where-does-the-money-comes-from/

- **DAFFARA,** C. (2012) "Estimating the Economic Contribution of Open Source Software to the European Economy".The First Openforum Academy Conference Proceedings . Disponible en: http://www.openforumacademy.org/research/the-first-openforum-academy-conference-proceedings

- **DRIVE**, M. (2012) "Key issues for Open- Source Software". Garther Research. Disponible en: https://www.gartner.com/doc/1359127/key-issues-opensource-software-

- **EELLS**, R. y **WALTON**, C. (1969) "Conceptual Foundations of Business" Homewood: IL. RichardD. Irwin

- **EL EMAM**, K. y **KOUR**, A. (2008) "A replicated survey of IT Software Project failure" Software IEEE volumen 25 número 5

- **Free Software Foundation** (1991) "Gnu general public license". Disponible en: www.fdf.org/licenses/gpl.htlm

- **GARCÍA-GARCÍA**, J. y M.I. **ALONSO DE MAGDALENO** (2014). Comunicación de la responsabilidad social en el sector del software libre, Universia Business Review, No. 41, pp. 98-124.

- **GARCÍA-GARCÍA**, J. y M.I. **ALONSO DE MAGDALENO** (2013) "Valuation of Open Source Software: How do you put a value on free?" Revista de Gestão, Finanças e Contabilidade, vol. 3, No. 1, pp. 3-16.

- **GARTNER GROUP** (2011) "Overview of perferences and practices in the Adoption and Usage of Open Source Software" Disponible en: https://www.gartner.com/doc/1528219/survey-analysis-overview-preferences-practices

- **GONZÁLEZ-BAHARONA**, J. **SEONANE**, J. y **ROBLES**, G. (2003) "Introducción al software libre" Creado por Universitat Oberta de Catalunya Mater Oficial del Software Libre. Disponible en: http://ocw.uoc.edu/informatica-tecnologia-y-multimedia/introduccion-al-software-libre/materiales/

- **GUERRA** N. (2013) "Incompatible Free Software License". Revista Chilena de Derecho y Tecnología Volumen 2 número 1. Págs 169-195. Disponible en: www.rchdt.uchile.cl/index.php/RCHDT/article/abwnload/22402/28937

- **HANN**, IL-H., **ROBERTS**, J., **SLAUGHTER**, S. y **FIELDING**, R.(2004) "An empirical analysis of economic returns to open source participation". Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.6697&rep=rep1&type=pdf

- **HERRERO**, J.J. (2010) "El software libre como herramienta de estrategia empresarial" *Chief Technologist* en tecnologías de software de Telefónica I+D, dentro del seminario sobre software libre en entorno empresarial organizado alrededor del máster en Economía Digital e Industrias Creativas.

- **Informe IRIA** (2012) "Las Tecnologías de la Información y las Comunicaciones en las Administraciones Públicas" Disponible en: http://administracionelectronica.gob.es/

- **Informe REINA** (2013) "Las Tecnologías de la Información y las comunidades en las Administraciones del Estado" Disponible en: http://administracionelectronica.gob.es/

- **LIPPOLDT**, D. y **STRYSZOWKI**, P. (2009) "Innovation in the Software Sector" OECD Innovation Strategy, Paris, Organization for Economic Cooperation and Development (OECD).

- **LUNDELL**, B., **MCKNIGHT**, J. y **GAHM**, J. (2010) "IT Spending survey" Research Report enero 2013. Disponble en: http://www.esg-global.com/default/assets/File/ESG%20Research%20Report%202010%20IT%20Spending%20Intentions%20Abstract.pdf

- **MAS**, J. (2005) "Marco jurídico y oportunidades de negocio en el software libre" Revista Sobre la Sociedad del Conocimiento número 1 (2005). Disponible en: http://www.uoc.edu/uocpapers/1/dt/esp/mas.pdf

- **MCQUAIDE**, B. (2010) "Distributed Multi-Source Developement whit Open Source" LinuxCon 2010
- **MOLLICK**, E.R. (2013) "The Dynamics of Crowdfunding: An Exploratory Study" Journal of Business Venturing, Volumen 29, Número 1, enero 2014, páginas 1-16.
- **MOODY**, G. (2002). Rebel code: Linux and the open source revolution. Basic Books, NY.
- Mozilla public license 1.1. Disponible en: http://www.mozilla.org/MPL/1.1/ Revisada: 26.02.2014
- **OECD** (2006) "The software sector: a statistical profile for selected oecd countries" Policy Brief, July 2006
- **POLANCO**, M. y **TAIBO**, B. (2011) "Android, Él sistema operativo de Google para dispositivos móviles" Negotium 2011 7(19). Disponible en: http://www.redalyc.org/pdf/782/78219156004.pdf
- **RAYMOND**, E.S. (1997) "How To Become a Hacker". Disponible en: http://www.catb.org/esr/faqs/hacker-howto.html
- **RAYMOND**, E. S. (2001). The Cathedral & the Bazaar: Musings on linux and open source by an accidental revolutionary. O'Reilly Media: Beijing, China.
- **RIEHLE**, D. (2007) "The Economic Motivation of Open Source Software: Stakeholder Perspective" *IEEE Computer* , vol. 40, no.4 (abril de 2007). Página 25-32.
- **RIEHLE**, D., **BERSACHNERDER**, S. (2012) "A Model of open Source Developer Foundation". Disponible en: dirkriehle.com/up_content/uplads/2012/05/Riehle_MOSDF-V12-Final-Web.pdf
- **ROBLES**, G. **SCHERDER**, H. **TRETKOWSKI**, I. **WEBER**, N. (2001) "Who is doing it". Disponible en: www.univ-paris13.fr/cepn/IMG/pdf/wp2007_16.pdf

- **ROBLES**, G. **GONZÁLEZ**, J.M. **CENTENO-GONZÁLEZ**, J. **MATELLÁN-OLIVERO**, V. y **RODENO-MERIN**, L. (2003) "Studying the evolution of free software project using publicly available data" Universidad Rey Juan Carlos. Disponible en: http://flosshub.org/system/files/111-115.pdf

- **SCHMIDT**, E. (2009) "Android Adoption is About To Explode" Disponible en: http://techcrunch.com/2009/10/15/schmidt-android-adoption-is-about-to-explode

- **STALLMAN**. R. (2002) "Software libre para una sociedad libre" Free Software Foundation

- **UNU- MERIT y COMICIÓN EUROPEA** (2006) Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU.

- **VAN HIPPEL**, E. y **VON KROGH**, G. (2003) "Open source software and the private" Collective innovation model

- **WITSA** "Digital planet " (2010). Disponible en: http://www.witsa.org/v2/media-center/pdf/DP2010_ExecSumm_Final_LbRes.pdf

- W3Counter. www.w3counter.com. Revisada: 07.05.2013