novática
CEPIS UPGRADE Revista de la Asociación de Técnicos de Informática

Novática, revista fundada en 1975 y decana de la prensa informática española, es el órgano oficial de expresión y formación continua de **ATI** (Asociación de Técnicos de Informática), organización que edita también la revista **REICIS** (Revista Española de Innovación, Calidad e Ingeniería del Software). ***Novática*** edita asimismo **UP**GRADE, revista digital de **CEPIS** (*Council of European Professional Informatics Societies*), en lengua inglesa, y es miembro fundador de **UP**ENET (**UP**GRADE *European* ***NET****work*).

<http://www.ati.es/novatica/>
<http://www.ati.es/reicis/>
<http://www.upgrade-cepis.org/>

ATI es miembro fundador de **CEPIS** (*Council of European Professional Informatics Societies*) y es representante de España en **IFIP** (*International Federation for Information Processing*); tiene un acuerdo de colaboración con **ACM** (*Association for Computing Machinery*), así como acuerdos de vinculación o colaboración con **AdaSpain**, **AI2, ASTIC, RITSI** e **Hispalinux**, junto a la que participa en **ProInnova**.

Coordinación Editorial, Redacción Central y Redacción ATI Madrid
Padilla 66, 3º, dcha., 28006 Madrid
Tlfn.914029391; fax.913093685 <novatica@ati.es>
Composición, Edición y Redacción ATI Valencia
Av. del Reino de Valencia 23, 46005 Valencia
Tlfn./fax 963330392 <secreval@ati.es>
Administración y Redacción ATI Cataluña
Via Laietana 46, ppal. 1ª, 08003 Barcelona
Tlfn.934125235; fax.934127713 <secregen@ati.es>
Redacción ATI Aragón
Lagasca 9, 3-B, 50006 Zaragoza.
Tlfn./fax 976235181 <secreara@ati.es>
Redacción ATI Andalucía <secreand@ati.es>
Redacción ATI Galicia<secregal@ati.es>
Suscripción y Ventas <http://www.ati.es/novatica/interes.html>, ATI Cataluña, ATI Madrid
Publicidad
Padilla 66, 3º, dcha., 28006 Madrid
Tlnf.914029391; fax.913093685 <novatica@ati.es>
Imprenta: Derra S.A., Juan de Austria 66, 08005 Barcelona.
Depósito legal: B 15.154-1975 -- ISSN: 0211-2124; CODEN NOVAEC
Portada: Cayendo hacia arriba - Concha Arias Pérez / © ATI
Diseño: Fernando Agresta / © ATI 2003

Nº 199, mayo-junio 2009, año XXXV

sumario

editorial

Comunicado de Prensa de ATI sobre la creación del Consejo de Colegios de Ingeniería Informáticas (17/6/2009) > 02

en resumen

Un mundo dinámico donde lo importante es sumar > 02
Llorenç Pagés Casas

IFIP

Reunión del TC6 (Communication Networks) > 03
Ramón Puigjaner Trepat

Reunión del TC1 (Foundations of Computer Science) > 03
Michael Hinchey, Joaquim Gabarro Vallés

monografía

Software libre para empresas

(En colaboración con **UP**GRADE*)*
Editores invitados: *Jesús M. González-Barahona, Teo Romera Otero y Böjrn Lundell*

Presentación. Software libre para empresas: ¡crea tu producto, alimenta tu comunidad y disfruta tu parte! > 04
Jesús M. Gonzalez-Barahona, Teófilo Romera Otero, Björn Lundell

El software libre en el mundo corporativo > 08
Jesús M. Gonzalez-Barahona, Teófilo Romera Otero, Björn Lundell

Buenas prácticas para la adopción del software libre > 12
Carlo Daffara

Construir y mantener una comunidad de práctica: método aplicado a proyectos de software libre > 17
Stéphane Ribas, Michel Cezon

Dinamización de comunidades en proyectos de software libre > 21
Martin Michlmayr

La Comunidad Morfeo: estrategias Open Source para la Open Innovation > 25
Cristina Breña, Andrés Leonardo Martínez Ortiz

Aplicación de los principios del software libre en líneas de producto > 29
Frank van der Linden

Abordar las necesidades de la industria en Software Libre > 36
Jan Henrik Ziesing

SpagoWorld, la iniciativa de software libre de Engineering > 39
Gabriele Ruffatti

Una oportunidad para las empresas de software libre: mercado emergente en los países en vías de desarrollo > 44
Susana Muñoz Hernández, Jesús Martínez Mateo

secciones técnicas

Enseñanza Universitaria de la Informática

Un juego de rol para la enseñanza de la profesión informática > 47
Agustín Cernuda del Río, Manuel Quintela Pumares, Miguel Riesco Albizu

Gestión del Conocimiento

Datos, conocimiento, información... en este orden > 51
Joan Baiget Solé

Informática Gráfica

Tratamiento de geoinformación a nivel peatonal: animación 3D a partir de metainformación Exif > 53
Ricardo Navarro Moral, Francisco R. Feito Higueruela, Rafael J. Segura Sánchez, Angel L. García Fernández

Tecnologías para la Educación

Influencia en el rendimiento académico de la interacción en línea de los alumnos: estudio y análisis comparativo entre diferentes modalidades de enseñanza > 59
Ángel Hernández García, Santiago Iglesias Pradas, Julián Chaparro Peláez, Félix Pascual Miguel

Referencias autorizadas > 63

sociedad de la información

Prospectiva Tecnológica

Un metaanálisis de la actividad prospectiva internacional en el campo de los Sistemas y Tecnologías de la Información > 70
Víctor Amadeo Bañuls Silvera, José Luis Salmerón Silvera

Personal y transferible

Una paradoja divina > 73
Antonio Vaquero Sánchez

Programar es crear

Reconstrucción (CUPCAM 2007, problema G, solución) > 75
Enrique Martín Martín, Cristóbal Pareja Flores

asuntos interiores

Coordinación Editorial / Programación de Novática / Socios Institucionales > 77

Monografía del próximo número: "*Calidad del software*"

Presentación
Software libre para empresas: ¡crea tu producto, alimenta tu comunidad y disfruta tu parte!

Jesús M. Gonzalez-Barahona[1], Teófilo Romera Otero[1], Björn Lundell[2]
[1]Universidad Rey Juan Carlos, Madrid; [2]Centro de Investigación en Informática de la Universidad de Skövde, Suecia

<{jgb,teo}@gsyc.es>, <bjorn.lundell@his.se>

En los últimos años la relevancia que el software libre ha ido ganando en diversos ámbitos y niveles no ha hecho más que crecer. Está presente en los mercados, en las administraciones públicas y es objeto de estudio y valoración por parte de académicos, empresas e instituciones de gobierno. Se modelan procesos de desarrollo de software siguiendo pautas nacidas en las comunidades de software libre, se diseñan modelos de negocio para empresas a partir de actividades típicas de los proyectos de software libre, los partidos políticos comienzan a preparar acciones al respecto del software libre en sus propuestas electorales y los gobiernos diseñan leyes respecto a su uso. E incluso se estudian modelos económicos alternativos en base a las características que han hecho posible una comunidad fuerte con capacidad para coordinar profesionales, empresas, voluntarios e instituciones públicas alrededor de productos de calidad y disponibles para cualquiera.

Mas allá del software, la cultura libre influye sin duda y a velocidad de vértigo en diversas facetas de las tecnologías de la información, el arte, la cultura y en definitiva en nuestras vidas. Elementos como la Wikipedia, la innovación abierta, la meritocracia o las licencias alternativas para el arte y la cultura basadas en los fundamentos introducidos por el software libre (como las licencias Creative Commons), las comunidades para compartir y crear en Internet como Flickr, y tantas otras novedades han sido absorbidas por nuestra sociedad líquida (que diría Zygmunt Bauman) con pasmosa naturalidad y a un ritmo imparable.

Los profesionales de las tecnologías de la información y la comunicación, así como las empresas del sector no pueden pasar por alto estas nuevas tendencias y tanto es así que resulta difícil encontrar una sola de estas empresas que no tenga ya una estrategia clara respecto al software libre, bien sea para competir con él, para adoptarlo en sus modelos de negocio o para dedicarse en exclusiva a su creación y explotación.

Este número monográfico especial ha sido confeccionado contactando con diferentes expertos profesionales e investigadores del mundo de la empresa, del software libre y de las relaciones entre empresa y comunidades de software libre. No pretendemos que sea una recopilación exhaustiva pero sí representativa del panorama actual en Europa en lo que se refiere a casos descriptivos de cómo las empresas usan y crean software libre, y especialmente cómo se relacionan con otros actores (otras empresas, voluntarios y profesionales) dentro del mismo escenario.

Editores invitados

Jesús M. Gonzalez-Barahona es profesor e investigador en la Universidad Rey Juan Carlos de Móstoles (Madrid). Comenzó a involucrarse en software libre en 1991. Desde entonces ha colaborado en numerosos grupos de trabajo, ha desarrollado algunas líneas de investigación y ha comenzado programas de formación diversos en la materia. También colabora con varios proyectos y asociaciones de software libre, escribe en diversos medios sobre temas relacionados con el software libre y lleva a cabo consultoría para compañías y administraciones públicas sobre software libre, todo ello desde el marco del grupo de investigación GSyC/LibreSoft, <http://libresoft.es>.

Teófilo Romera Otero es Ingeniero en Informática por la Universidad Rey Juan Carlos de Madrid, donde está actualmente llevando a cabo tareas de investigación para la consecución de su doctorado. Sus intereses académicos son el software de fuentes abiertas y cómo las empresas se relacionan con el. También está interesado en el desarrollo de software global aplicado al desarrollo de software de fuentes abiertas y la habilitación tecnológica a través de software de fuentes abiertas. Trabaja para el grupo de investigación GSyC/LibreSoft como coordinador de proyectos de investigación nacionales y europeos, algunos de ellos: Calibre, Edukalibre, FLOSSWorld, Morfeo, Vulcano, QualiPSo, FLOSSInclude o Tree. Otras tareas que lleva a cabo en el grupo son consultorías técnicas sobre software de fuentes abiertas y la coordinación docente del *Master on Libre Software* organizado por GSyC/LibreSoft. Es miembro del *Qualipso Network Board* que gestiona la red de centros de competencia Qualipso y del grupo de trabajo sobre Software Libre de la plataforma NESSI. Ha realizado estancias de investigación en la universidad de Leeds en el Reino Unido y en el *Irish Software Engineering Research Centre* (Lero) de la universidad de Limerick (Irlanda).

Björn Lundell es investigador en el Centro de Investigación en Informática de la Universidad de Skövde. Sus intereses académicos incluyen el software de fuentes abiertas, su evaluación y el soporte de métodos. Fue gestor técnico del proyecto de investigación COSI (*Codevelopment Using Inner & Open Source in Software Intensive Products*). Es miembro fundador del grupo de trabajo IFIP 2.13 sobre software libre y preside la asociación industrial *Open Source Sweeden*. Lundell es doctor en Ciencias de la Computación por la Universidad de Exeter.

En un primer artículo introductorio, *El software libre en el mundo corporativo*, los editores invitados del presente número, ***Jesús M. González Barahona***, ***Teófilo Romera Otero*** y ***Björn Lundell***, ofrecen su visión particular de la importancia de que el mundo empresarial tenga estrategias de acción respecto del software libre. Al mismo tiempo, sirve de introducción a los nuevos términos y conceptos a los que se hace referencia en el resto del número, intentando ofrecer una mirada a vista de pájaro de los motivos y los medios que aplican en este nuevo mundo que es el software libre para empresas.

En su artículo, *Buenas prácticas para la adopción del software libre*, ***Carlo Daffara***, reconocido experto en modelos de negocio sostenibles con software libre, recoge un compendio de consejos y directrices que aseguran una migración al software libre exitosa para entornos empresariales. Este artículo abre el monográfico porque resulta muy representativo de los problemas que las empresas suelen encontrar a la hora de incluir el software libre en sus procesos o en sus modelos de negocio y al mismo tiempo puede servir al lector no iniciado como una extensión de la introducción, puesto que maneja aspectos técnicos que resultan familiares al profesional del sector.

Seguidamente, el artículo *Construir y mantener una comunidad de práctica: método aplicado sobre proyectos de software libre* muestra un método para la creación y gestión de comunidades que puede ser aplicado a la creación y gestión de comunidades de software libre y especialmente a aquellas que

como OW2 o Morfeo implican relaciones entre voluntarios, profesionales y empresas. Los autores, ***Stéphane Ribas*** y ***Michel Cezon***, conocen bien tanto la comunidad del software libre como el mundo empresarial y tras su paso por diversas empresas informáticas han recalado recientemente en el mundo académico, para desde el instituto INRIA seguir ayudando a construir buenas relaciones entre comunidades de desarrollo y empresas.

En línea con el artículo anterior, ***Martin Michlmyer*** nos aporta su trabajo *Dinamización de comunidades en proyectos de software libre*. Se trata de un artículo que complementa muy bien tanto al artículo anterior, sobre la construcción de comunidades, como al que le sigue. Sin duda, se trata de un tema muy interesante y novedoso, puesto que aporta una descripción formal de la figura del dinamizador de comunidades en las grandes empresas. Un fenómeno reciente de importancia creciente y relativamente poco conocido hasta el momento. No en vano, Martin ha sido en parte su propio objeto de estudio y observación puesto que su trayectoria como líder de diversos proyectos de software libre de gran envergadura e importancia (incluyendo Debian) le ha llevado a ser contratado por HP para dinamizar y gestionar las relaciones entre la empresa y las comunidades de software libre.

Con una fuerte relación con los dos artículos anteriores, ***Andrés Leonardo Martínez Ortíz*** y ***Cristina Breña***, co-fundador y responsable de comunicación respectivamente de la comunidad Morfeo liderada por Telefónica I+D, presentan *La Comunidad Morfeo: estrategias Open Source para la Open Innovation*. Morfeo es una comunidad de software en la que participan actores de muy diversa índole y con intereses y capacidades dispares, y aun así exitosa. Se trata pues de un caso de estudio descriptivo de la historia y peculiaridades de la comunidad Morfeo.

Una vez cubierto el tan interesante tema de las relaciones entre comunidades de desarrollo de software libre y empresas, el número continúa con algunos otros trabajos menos orientados a la creación y mantenimiento de comunidades pero mucho más enfocados a la descripción de casos de éxito, oportunidades y estrategias que las empresas siguen ya o tienen a su disposición para la adecuada explotación de nuevos modelos de negocio basados en software libre.

Desde su amplia experiencia en Philips, ***Frank van der Linden***, reconocido experto en software libre y líneas de producto, escribe su artículo *Aplicación de los principios del software libre en líneas de producto* en el que investiga las diversas opciones en las que el software libre y las metodologías de desarrollo asociadas pueden utilizarse para reducir los problemas del desarrollo de software global (GSD o desarrollo distribuido) así como para aumentar la calidad del software que se desarrolla.

Jan Henrik Ziesing, científico investigador del Instituto Fraunhofer en sistemas de comunicación abiertos en Berlín, aporta un trabajo sobre las necesidades de la industria en el ámbito de software libre y la participación del Instituto Fraunhofer FOKUS en el proyecto Qualipso, el mayor proyecto del programa marco de la Comisión Europea de los que tratan exclusivamente con el software libre. El artículo lleva por título *Abordar las necesidades de la industria en Software Libre*. Muestra la estrategia que Fraunhofer FOKUS tiene preparada para llevar sus modelos y servicios de interoperabilidad tradicionales al ámbito del software libre, mediante la creación de un nuevo centro de competencia del software libre que ofrezca servicios al tejido empresarial de la región de Berlín.

En el siguiente artículo, *SpagoWorld, la iniciativa de software libre de Engineering*, ***Gabriele Ruffati***, director de la unidad de Arquitecturas y Consultoría de la división de I+D en Engineering, muestra un caso de estudio basado en la experiencia de la compañía. Se describen las razones, la estrategia y las relaciones de Engineering con las comunidades, así como los resultados obtenidos por la empresa en sus productos libres.

Por último y para cerrar el número, ***Susana Muñoz Hernández*** y ***Jesús Martínez Mateo***, profesora y estudiante de doctorado respectivamente, de la Universidad Politécnica de Madrid (UPM) e investigadores ambos del grupo de cooperación TEDECO (TEcnología para el DEsarrollo y la COoperación), presentan el artículo *Una oportunidad para las empresas de software libre: mercado emergente en los países en vías de desarrollo*. En él se describen oportunidades y mecanismos mediante los cuales el software libre puede ayudar a la creación del tejido empresarial en los países en vías de desarrollo.

En definitiva, el presente número especial *Software libre para empresas* pretende mostrar un panorama actual de las iniciativas, estrategias y acciones que las empresas europeas están llevando a cabo respecto al software libre. Hemos intentado recabar un conjunto de artículos equilibrado y de fuentes y procedencias diversas, incluyendo Italia, Francia, Holanda, Alemania y España, con autores de renombre que se sitúan en la franja, cada vez más amplia, en la que se solapan las comunidades de software libre y el mundo empresarial, y ofreciendo puntos de vista por parte de universidades, grandes empresas, individuos y organismos públicos. Esperamos que el resultado sea del agrado del lector y les emplazamos a seguir las evoluciones futuras de la presente publicación.

Agradecimientos

Los editores invitados de este número especial de ***Novática*** quieren agradecer al equipo de ATI y en especial a Llorenç Pagés, por su ayuda y buen hacer. También agradecer a Miguel Vidal su ayuda como coordinador de las traducciones y a los traductores del grupo GSyC/LibreSoft por su magnífico trabajo.

Referencias útiles sobre "Software Libre en la empresa"

Las referencias que se citan a continuación, junto con las proporcionadas en cada uno de los artículos, tienen como objetivo ayudar a los lectores a profundizar en los temas tratados en esta monografía permitiendo contrastar ideas y obtener información actualizada.

Libros

■ **Dan Woods, Gautam Guliani.** *Open Source for the enterprise*. O'Reilly Media, Inc. (2005). ISBN-10: 0596101198. Este es un gran libro que explica el software libre desde el punto de vista empresarial. Ofrece ideas brillantes sobre la gestión de proyectos y la paquetización de software libre.

■ **Jan Sandred.** *Managing Open Source Projects*. John Wiley & Sons, 2001. Un manual sobre cómo abordar los principios y ventajas de la programación de software libre.

■ **Jesús González Barahona, Joaquín Seoane Pascual, Gregorio Robles.** *Introducción al Software Libre*. Es un libro de texto usado en el tema de Software Libre del programa de doctorado de la Universidad Rey Juan Carlos y en algunos otros programas de la *Universitat Oberta de Catalunya*. Cubre prácticamente todos los temas relacionados con el software libre. Este es un libro imprescindible. <http://ocw.uoc.edu/computer-science-technology-and-multimedia/introduction-to-free-software/materials/>.

■ **Karl Fogel.** *Producing Open Source Software: How to Run a Successful Free Software Project*. O'Reilly Media, Inc. (2005). ISBN-10: 0596007590. Este es un libro sobre el aspecto humano del desarrollo de software

libre. Describe cómo operan los proyectos exitosos, las expectativas de usuarios y desarrolladores, y la cultura del software libre <http://producingoss.com/>.

■ **Chris DiBona, Sam Ockman, Mark Stone et al.** *Open Sources: Voices from the open source revolution.* O'Reilly Media, Inc. (1999). ISBN-10: 1565925823. Una clásica y excelente compilación de artículos, ensayos y escritos de importantes actores en la escena del software libre. Entre ellos se encuentran Linus Torvalds, Eric S. Raymond, Richard Stallman, Bruce Perens y muchos otros <http://oreilly.com/catalog/opensources/book/toc.html>.

■ **Donald K Rosenberg.** *Open Source: The Unauthorized White Papers.* Hungry Minds, 2000. ISBN-10: 0764546600. Se trata de una vision general del software libre <http://www.stromian.com/Book/FrontMatter.html>.

■ **Lawrence Rosen.** *Open Source Licensing: Software Freedom and Intellectual Property Law.* Prentice Hall PTR, 2004. ISBN-10: 0131487876. Este libro trata de casi todo lo que hay que saber sobre las licencias de software libre <http://www.rosenlaw.com/oslbook.htm>.

■ **Lawrence Lessig.** *Free Culture.* Penguin Press HC, 2004. ISBN-10: 1594200068. Este libro explica cómo los grandes medios de comunicación usan la tecnología y las leyes para bloquear la cultura y controlar la creatividad <http://free-culture.org/>.

■ **Karl Fogel, Moshe Bar.** *Open Source Development with CVS.* Paraglyph Inc. (2003). ISBN-10: 1932111816. Explica muchos aspectos del software libre a través de la descripción de técnicas y herramientas de desarrollo. <http://cvsbook.red-bean.com/>.

Otras publicaciones

■ **Qualipso Project.** Qualipso deliverable on OSS business models (2008). Explica muchos aspectos de los modelos de negocio del software libre junto con el estado del arte actual <http://qualipso.org/sites/default/files/media/A2/A2.D1.2.3%20The%20business%20models%20for%20using%20OS.pdf>.

■ **A. Abella, M. A. Segovia.** *Libro blanco del software libre en España* (2007). <http://libroblanco.com/document/III_libro_blanco_del_software_libre.pdf>.

■ *2020 FLOSS Roadmap.* Esta "hoja de ruta" fue presentada en el OpenWorldForum 2008 en Paris. En un documento muy útil creado de forma colaborativa por un gran conjunto de expertos de diferentes empresas y entidades <http://www.2020flossroadmap.org/>.

■ **Rishab Aiyer Ghosh et al.** *Study on the: Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU (FLOSSImpact Report 2006).* Este informe fue elaborado en el marco del proyecto FLOSSImpact <http://www.flossimpact.eu/> y una gran cantidad de investigadores participaron en él <http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf>.

■ **Qualipso Project.** *Analysis of most important aspects of Open Source Competence Centres.* <http://qualipso.org/node/48>.

El software libre en el mundo corporativo

Jesús M. Gonzalez-Barahona[1], Teófilo Romera Otero[1], Björn Lundell[2]

[1]Universidad Rey Juan Carlos, Madrid; [2]Centro de Investigación en Informática de la Universidad de Skövde, Suecia

<{jgb,teo}@gsyc.es>,
<bjorn.lundell@his.se>

Resumen: *el software libre es un nuevo mundo en sí mismo tanto para las empresas como para los profesionales. Es por eso que comprender cómo encontrar oportunidades en él es cada vez más importante a medida que los productos de software libre se usan más y más en la industria del software y en otros sectores que dependen del software para sus actividades. En este texto, se exploran algunos de los nuevos aspectos que aparecen al aproximarse al software libre y cómo las empresas reaccionan a ellos.*

Palabras clave: *comunidades, empresas, estrategia, FLOSS, OSS, software libre.*

Autores

Jesús M. Gonzalez-Barahona es profesor e investigador en la Universidad Rey Juan Carlos de Móstoles (Madrid). Comenzó a involucrarse en software libre en 1991. Desde entonces ha colaborado en numerosos grupos de trabajo, ha desarrollado algunas líneas de investigación y ha comenzado programas de formación diversos en la materia. También colabora con varios proyectos y asociaciones de software libre, escribe en diversos medios sobre temas relacionados con el software libre y lleva a cabo consultoría para compañías y administraciones públicas sobre software libre, todo ello desde el marco del grupo de investigación GSyC/LibreSoft, <http://libresoft.es>.

Teófilo Romera Otero es Ingeniero en Informática por la Universidad Rey Juan Carlos de Madrid, donde está actualmente llevando a cabo tareas de investigación para la consecución de su doctorado. Sus intereses académicos son el software de fuentes abiertas y cómo las empresas se relacionan con el. También está interesado en el desarrollo de software global aplicado al desarrollo de software de fuentes abiertas y la habilitación tecnológica a través de software de fuentes abiertas. Trabaja para el grupo de investigación GSyC/LibreSoft como coordinador de proyectos de investigación nacionales y europeos, algunos de ellos: Calibre, Edukalibre, FLOSSWorld, Morfeo, Vulcano, QualiPSo, FLOSSInclude o Tree. Otras tareas que lleva a cabo en el grupo son consultorías técnicas sobre software de fuentes abiertas y la coordinación docente del *Master on Libre Software* organizado por GSyC/LibreSoft. Es miembro del *Qualipso Network Board* que gestiona la red de centros de competencia Qualipso y del grupo de trabajo sobre Software Libre de la plataforma NESSI. Ha realizado estancias de investigación en la universidad de Leeds en el Reino Unido y en el *Irish Software Engineering Research Centre* (Lero) de la universidad de Limerick (Irlanda).

Björn Lundell es investigador en el Centro de Investigación en Informática de la Universidad de Skövde. Sus intereses académicos incluyen el software de fuentes abiertas, su evaluación y el soporte de métodos. Fue gestor técnico del proyecto de investigación COSI (*Codevelopment Using Inner & Open Source in Software Intensive Products*). Es miembro fundador del grupo de trabajo IFIP 2.13 sobre software libre y preside la asociación industrial *Open Source Sweeden*. Lundell es doctor en Ciencias de la Computación por la Universidad de Exeter.

1. El concepto de software libre, (free, open source) software

Software libre es el término que utilizamos en este artículo para referirnos tanto al "*free software*" según lo define la Free Software Foundation[1], como al "*open source software*" (software de fuentes abiertas) según lo define la Open Source Initiative[2]. Aunque ambas definiciones son diferentes, cubren casi la misma colección de software. En cualquier caso, los términos (al menos en inglés) no son intercambiables. Para algunos, la palabra "free" (que se puede traducir como libre o como gratis al castellano) se refiere a las libertades que el software libre brinda y por esto mismo, nunca debería ser omitida. Para otros, el sentido ambiguo de la palabra "free" que puede entenderse también como "gratis" es un problema para el entendimiento del concepto, especialmente en entornos empresariales. Para evitar esta discusión, usaremos el término software libre a lo largo de este texto.

En cualquier caso, podemos aportar una breve descripción del software libre mencionando las "cuatro libertades". El software libre es aquel que permite a los que lo reciben: ejecutarlo y usarlo; estudiarlo y adaptarlo a necesidades específicas; redistribuirlo a otros; y mejorarlo y añadirle funcionalidad. Todas estas libertades pueden ejercerse una vez que el software es obtenido, sin necesidad de permisos adicionales de los dueños del copyright más allá de la licencia que se aplica para ese software.

Desde este punto de vista, el software libre es principalmente un concepto legal. Define algunos permisos básicos que los dueños del copyright garantizan a aquellos a quienes dan el software. De hecho, los dueños del copyright no ceden todos sus derechos. Algunos quedan reservados en forma de cláusulas y condiciones especificadas en las licencias de software libre.

Desde un punto de vista económico y de negocio, es importante remarcar que no se especifica nada acerca de cómo se obtiene el software. Puede ser obtenido gratis de un repositorio público en Internet o comprado por un sustancioso precio en una tienda, puede que en una bonita caja.

Desde un punto de vista técnico, el software libre no es una tecnología en sí mismo. De hecho, la única característica tecnológica común para que un programa sea software libre, es que el código fuente ha de estar disponible (el término "software de fuentes abiertas" hace hincapié en esto). Pero aparte de este hecho, existen algunas particularidades comunes en la manera en que se desarrolla el software libre: modelos de desarrollo abiertos, con más información disponible públicamente; existencia de comunidades de soporte; mezcla de voluntarios y desarrolladores pagados; meritocracia y reglas de gobierno basadas en la comunidad, etc.

Desde un punto de vista ético, la ética *hacker* es una fuerte convicción para muchos de los participantes en los proyectos de software libre, pero no se trata de un aspecto "obligatorio". De hecho, muchos otros no la reconocerán o incluso no saben de qué se trata.

Puede que esta breve introducción a la idea de software libre sea suficiente para mostrar cómo de complejo es este fenómeno, con implicaciones legales, de negocio, técnicas y éticas. Debido a esta naturaleza diversa y las interrelaciones de los diferentes intereses de la gente que participa, no suele ser fácil de entender para los neófitos.

2. Problemas comunes que encuentran las empresas

En algún momento una empresa, que puede estar enfocada a las TIC o bien utilizarlas para otros propósitos, considera el producir, integrar, usar o mejorar servicios basados en software libre. Existen varias razones que pueden llevarla a este punto. Razones ampliamente discutidas en otros lugares y que no serán consideradas aquí. En lugar de centrarse en esas razones, este artículo toma esa

decisión de "pasarse al software libre", que cada vez es más común en el mundo empresarial, como el punto de partida.

De hecho, una vez que se ha tomado esa decisión es cuando viene la parte más difícil. Las empresas han de lidiar, a partir de ese punto, con algunos aspectos que si son manejados con acierto pueden producir beneficios considerables, pero si son pasados por alto, pueden también ser causa de numerosos riesgos y problemas. Muchos de estos aspectos son someramente discutidos en el resto de esta sección.

2.1. Lidiar con comunidades

El desarrollo de software libre se suele llevar a cabo en comunidades que no sólo incluyen a los desarrolladores principales, sino también a colaboradores esporádicos o desarrolladores de productos relacionados (por ejemplo de ramas alternativas del mismo software, para el uso corporativo) y usuarios de diferentes tipos (desde usuarios finales que no saben nada del desarrollo a integradores con experiencia que conocen el producto casi tan bien como los desarrolladores principales). Los individuos en estas comunidades pueden ser simplemente voluntarios que dependen de su propia motivación, profesionales contratados por empresas para contribuir de acuerdo a los intereses de estas o incluso alguna combinación de ambos.

Cualquier compañía que se acerque (o que pretenda crear) a una de estas comunidades se enfrentará a un entorno bastante diferente de las relaciones contractuales que mantiene con otras compañías o empleados y a las que está acostumbrada. Por ejemplo, normalmente no hay contratos que regulen las relaciones entre participantes en la comunidad y que puedan comprometer a la gente una vez firmados. Al contrario, la motivación y la creación de relaciones fructíferas para todas las partes es aquí el modus operandi. Adaptarse a estas nuevas reglas no es siempre fácil.

La imagen corporativa también adquiere una nueva dimensión en estos entornos. Cuando una empresa (generalmente a través de sus empleados) se convierte en miembro de una comunidad de software libre, los desarrolladores comienzan a forjarse una imagen de la empresa, que normalmente se sobrepone a la imagen "tradicional" de dicha empresa. Otros comienzan a considerarla en términos de cuánto esa empresa contribuye a la comunidad, cuánto se beneficia de ella, cómo colabora con otros participantes dentro de ella o si es percibida como si estuviera intentando controlarla. Esta imagen puede producir todo tipo de efectos secundarios.

2.2. Asuntos legales

Las licencias de software libre, aunque se basan en las mismas leyes de propiedad intelectual con las que las empresas de TIC están familiarizadas, son utilizadas de modos muy diferentes. Lamentablemente, sigue siendo poco común encontrar expertos legales con experiencia y conocimientos plenos en los modelos de licenciamiento libres. Pero analizar los esquemas de licenciamiento con acierto y comprender no sólo sus implicaciones legales, sino también las económicas o las técnicas, es de crucial importancia para las compañías. Los esquemas de desarrollo, la diseminación de nuevas tecnologías o los modelos de negocio se ven normalmente restringidos o potenciados dependiendo de las licencias que entren en juego.

De hecho, las licencias de software libre son de algún modo el tejido que mantiene unidos a todos los actores interesados en un producto de software libre. Dependiendo de la licencia utilizada, de cómo es explicada y cómo se hace cumplir, otros actores podrán estar más o menos interesados en colaborar. Las licencias también brindan una serie de garantías a empresas interesadas en un producto y tienen impacto en muchos aspectos diferentes incluyendo la reutilización de componentes o la dependencia estratégica. Por supuesto que también muestran una fuerte relación con otros aspectos legales que preocupan a las compañías, como las leyes reguladoras de la propiedad industrial (patentes).

A pesar de la importancia de las licencias en el mundo del software libre, su estudio detallado no es siempre simple. De hecho, es un campo que el mundo corporativo suele reconocer como potencialmente problemático.

2.3. Nuevos procesos de desarrollo

El software libre puede ser desarrollado usando métodos tradicionales, que son comunes en las empresas de desarrollo de software, pero este no suele ser el caso. Las comunidades de software libre suelen utilizar sus propios procesos, que tienen en cuenta o incluso aprovechan las características particulares del equipo de desarrollo. Una de las principales diferencias que normalmente encuentran las compañías es que los desarrolladores no están atados por contratos o trabajan para una única compañía.

Al contrario, las comunidades de desarrollo de software libre por regla general, están compuestas de una mezcla de voluntarios y profesionales contratados por diferentes empresas. Debido a esta mezcolanza, las relaciones entre desarrolladores, y entre las compañías y los desarrolladores, son bastante diferentes a las de los proyectos tradicionales, con una jerarquía clara de gestión y restricciones contractuales claras si varias empresas se involucran a un tiempo.

En la mayoría de los proyectos de software libre, la palabra clave es: motivación. Los desarrolladores no pueden ser "gestionados" en el sentido clásico de la palabra. Por contra, trabajan juntos, con procesos de toma de decisiones y planificación normalmente basados en el concepto de meritocracia y confianza mutua. Por supuesto esto no significa que no haya toma de decisiones, o que las opiniones y propuestas de todos los participantes se tengan en consideración en la misma medida. Los proyectos de software libre suelen seguir ciertas reglas, que en algunos casos existen por escrito (al menos parcialmente), pero pueden ser muy diferentes de aquellas que se utilizan en el mundo empresarial.

Este nuevo entorno suele ser difícil de entender para algunos desarrolladores de empresas, que puede ser que tengan muchos años de experiencia, pero que pueden ser desconocedores de las particularidades de los proyectos de software libre, como por ejemplo las posibilidades de que una contribución sea aceptada o no o cómo hacerla de la mejor manera para que lo sea, que es una de las cosas que puede ser difícil de entender para ellos.

Cuando una empresa adopta un producto de software libre o decide lanzar el suyo propio, este tipo de problemas han de ser tenidos en cuenta. De cualquier modo, hay diferentes estrategias para lidiar con ellos, desde contratar nuevo personal con experiencia en entornos de software libre hasta proporcionar formación al equipo existente, para que aprendan y experimenten el tipo de situaciones que se van a encontrar.

2.4. Impulsar un producto

Cuando una empresa lanza un nuevo producto de software libre, normalmente está interesada en mantener algún tipo de control sobre él y su evolución futura. En el caso del software propietario, este control se ve garantizado por los estrictos términos de las licencias. Pero si se trata de software libre, otros pueden trabajar en él también, mejorarlo (puede que de una manera en que a la empresa no le guste) y redistribuir nuevas versiones, proporcionar servicios basados en él, y puede que, con el tiempo, hacerse con el control del producto (siendo por ejemplo que distribuya la versión más conocida y usada).

Para evitar esto, las empresas pueden utilizar diferentes estrategias, pero no dejan de ser estrategias en un nuevo entorno para ellas. De hecho, están basadas en mantener su estatus de productor principal del producto, que normalmente depende de cuánto esfuerzo y recursos están dispuestas a dedicar al producto una vez ha sido distribuido. Es por esto que han de planear de antemano si no quieren arriesgarse a perder su posición privilegiada. La situación es más problemática cuando la empresa no es el único productor, sino que se ha unido a una comunidad de desarrollo.

Normalmente, son varias las compañías que compiten en ella por ser consideradas "cabeza del proyecto" Buscar situaciones sinérgicas que involucren a otros actores y entender como contribuir a la creación de una comunidad sana que asegure la viabilidad del proyecto al mismo tiempo que se mantenga un cierto grado de control sobre ciertas decisiones, es un juego difícil.

3. Nuevas estrategias para nuevas situaciones

El software libre es un mundo nuevo para las compañías. Es territorio desconocido no sólo para los que producen software como su principal línea de actividad, sino también para aquellos que necesitan software para sus productos o servicios. Esta última categoría, engloba a una enorme porción de todas las compañías, puesto que son estratégicamente más y más dependientes del software.

Por tanto, se necesitan nuevas estrategias y muchas compañías las están explorando activamente para beneficiarse del software libre, para usar software libre como medio para reforzar otros fines, o incluso como parte integral de la política de negocio de la compañía. En esta sección se muestran algunas de estas estrategias, las cuales en muchos casos vienen a describir nuevas maneras de colaborar con otros actores.

3.1. Comunidades de empresas

Las comunidades de empresas basadas en el software libre son una nueva forma de colaboración empresarial no basada en relaciones contractuales. El núcleo alrededor del cual se forma la comunidad suele ser un conjunto de proyectos de software libre que pueden estar interrelacionados o no. Las empresas participan en la comunidad porque tienen algún tipo de interés en el software (producirlo, usarlo, proveer servicios alrededor de él, etc.), o incluso en la comunidad misma, por las oportunidades de negocio que pueden surgir de ella. De alguna manera, estas comunidades están modeladas según las comunidades tradicionales de software libre pero con la diferencia principal de que están compuestas por compañías en lugar de por individuos. No suelen estar sujetas a alianzas institucionales formales, aunque en algunos casos existe el concepto explícito de membresía. Los mismos principios de meritocracia y "los que más contribuyen son los que deciden" que se encuentran en otras comunidades de software libre, también se ponen en funcionamiento aquí , hasta cierto punto.

Estas comunidades de empresas son en parte similares a los grupos de interés informales pero con un lazo especial que armoniza los diferentes intereses: el código que se produce en la comunidad. En este marco, las empresas interesadas en obtener un producto concreto simplemente empiezan a desarrollarlo, puede que tras una colaboración pactada con otros actores. Las acciones de negocio pueden ser coordinadas por varios miembros y en algunos casos se encuentra cierto grado de aprovisionamiento común de fondos y recursos para lanzar nuevas ideas. Normalmente se trata de una mezcla de empresas grandes y pymes, junto con organismos públicos de investigación y otros entes. Esta peculiar mezcolanza facilita la transferencia tecnológica desde el mundo académico a la industria y la creación de ecosistemas ricos alrededor de proyectos de software libre específicos.

El hecho de que el software producido sea software libre garantiza que nadie tiene control absoluto sobre él y que se busquen soluciones con las que todos los diferentes actores se sientan a gusto. Esto de alguna manera allana el camino y ayuda a establecer algunas normas, lo que facilita el hecho de que compañías competidoras puedan, en algunas circunstancias, colaborar en proyectos específicos de su mutuo interés sin el engorro y la complejidad de detallados acuerdos legales. Algunos ejemplos de estas comunidades son OW2[3] (antiguamente Object Web), construida alrededor de tecnologías web, o Morfeo[4] que trata con diversas tecnologías.

3.2. Alianzas empresariales

Las alianzas son un caso específico de grupo de empresas con un objetivo específico y común a todos los participantes, que actúa como lazo de unión que las mantiene unidas. Este objetivo común es normalmente la producción o promoción de una tecnología específica o un estándar basado en el software libre. Las empresas participantes normalmente firman un acuerdo común e invierten recursos en ese fin. En general, no son muy diferentes de otros tipos de alianzas empresariales y sus principales diferencias son debidas al hecho de estar construidas alrededor del software libre. De hecho, esto suele ayudar a que la alianza se establezca como un punto neutral de control, abierto a nuevos actores. Ya que nadie controla la tecnología (puesto que es software libre), en principio el campo está allanado para que todos los participantes compitan según sus capacidades.

Algunos ejemplos de alianzas basadas en software libre son Limo Foundation[5] , Open Handset Alliance[6] , y Symbian Foundation[7] (todas en el área de los sistemas operativos para dispositivos móviles), o Genivi Alliance[8] (dedicada a la producción de una plataforma de software para coches).

3.3. Promoción de las comunidades tradicionales

Otra estrategia utilizada habitualmente por las empresas es la creación y promoción de una comunidad de desarrolladores y usuarios, diseñada según las comunidades de software libre tradicionales. En este caso, una compañía que lidera un proyecto de software libre intenta que otras compañías e individuos se interesen, usando una estructura basada en la meritocracia, la contribución y la confianza, de manera similar a lo que ocurre en las comunidades formadas por voluntarios.

La empresa que promueve la comunidad se posiciona en una figura de promotor benevolente, de algún modo renunciando a algunos de los beneficios de ser el principal productor. El hecho de que el producto sea software libre también proporciona ciertas garantías a otros actores de que el promotor no mantiene control total, lo que hace que sea más interesante para ellos el participar. Por regla general, se encuentran relaciones sinérgicas donde los promotores consiguen esfuerzo e innovación invertido en el producto en cantidades sustanciales, mientras que los actores se benefician bien de los recursos del promotor, bien de las oportunidades de negocio que se dan gracias a la escala de la colaboración.

Algunas empresas productoras de distribuciones de Linux, como RedHat[9] con su comunidad Fedora[10] o Canonical[11] con su comunidad Ubuntu[12] fueron de las primeras en explorar esta estrategia. Más recientemente, Nokia[13] ha sido un caso sonado también, con su comunidad Maemo[14] . Como estos casos ha habido muchos otros recientemente.

3.4. Involucrarse en comunidades tradicionales

En algunos casos, la mejor estrategia percibida por una empresa es unirse a una comunidad tradicional de desarrollo de software libre existente. En ocasiones, estas comunidades comenzaron como un grupo de individuos, probablemente voluntarios, con el objetivo común de compartir esfuerzos para el desarrollo y el mantenimiento de algún producto de software específico. Pero con el tiempo, es habitual que desarrolladores contratados por empresas interesadas en el producto acaben por unirse a la comunidad o que las empresas contraten directamente a desarrolladores de la comunidad, para de algún modo unirse al desarrollo colaborativo (y en algunos casos tratar de influenciarlo). Este ha sido el caso, por ejemplo, de una de las más conocidas comunidades de software libre: Apache[15] . Desde sus comienzos, varios desarrolladores clave han estado trabajando para compañías mientras participaban en Apache en tiempo de trabajo. La comunidad Apache reconoció este hecho pero trató de mantenerse como una comunidad de individuos en la cual los miembros no se consideran representantes de la empresa para la que trabajan. Aunque siempre puede haber algún tipo de tensión debido a intereses empresariales, de cuando en cuando.

En algún momento, algunas comunidades formalizan la participación de empresas. La comunidad acepta recibir donaciones (en forma de empleados, fondos u otros valores como infraestructuras, servidores, etc.) y se estipulan reglas para equilibrar los intereses de las empresas junto con los del proyecto como un todo.

Normalmente, las empresas tratan de impulsar el proyecto ya no requiriendo más influencia a cambio de sus donaciones, sino produciendo código para las funcionalidades específicas que están buscando. En algunos casos, esto lleva a la creación de nuevos productos dentro del proyecto. Esto puede distorsionar la relación entre voluntarios y empleados en el proyecto. Pero también produce muchos beneficios, uno de los cuales puede ser el incremento de los recursos humanos que trabajan activamente en el proyecto o la solución a problemas específicos en los que los

voluntarios podrían no estar tan interesados. Existen varios ejemplos de casos como este. El Proyecto GNOME[16] es probablemente el caso más conocido. A finales de los 90 y comienzos de esta década, muchas empresas (grandes corporaciones como Sun[17] o Novell[18], o pequeñas *startups* como Eazel[19] o Ximian[20]) colaboraron juntas. Un caso más moderno puede ser la relación de la Mozilla Foundation[21] con varias compañías, especialmente Google[22], de la que recibe una cantidad considerable de ingresos.

4. El papel de las pymes

Algunas pequeñas y medianas empresas tienden a ser muy activas en términos de innovación y creación de nuevas tecnologías. Normalmente son consideradas más ágiles y dinámicas que las grandes corporaciones. Es por esto que no es sorprendente que muchas de ellas hayan sido pioneras en el campo del software libre.

Algunos de los primeros casos se dieron gracias a desarrolladores de software libre que fundaban sus propias pymes para rentabilizar el tiempo y esfuerzo invertido en la creación de producto de software libre exitoso. En estos casos, el desarrollo fue comenzado por una comunidad de voluntarios, pero en algún momento, una parte o todos ellos, decidieron crear una empresa que probablemente se encargaba de todo o casi todo el desarrollo y ofrecía servicios con ánimo de obtener beneficios. Un ejemplo muy representativo de esto es el caso de MySQL[23], donde los principales desarrolladores fundaron MySQL AB, la empresa que lideró a partir de entonces el desarrollo casi al completo. Al mismo tiempo, ofrecieron servicios con los que obtenían fondos para financiar el desarrollo.

Este modelo se encuentra en muchos otros casos antes y después del de MySQL. De hecho, es tan popular que en muchos casos, la empresa se establece incluso antes de que el desarrollo comience formalmente, con el objetivo formal de dirigirlo cuando empiece. De algún modo, Red Hat o Ximian siguieron este modelo, siendo sólo dos de los muchos ejemplos existentes. Con el tiempo, estas compañías pueden llegar a ser compradas por grandes corporaciones (como ha sido el caso de MySQL AB, adquirida por SUN o Ximian, adquirida por Novell), que de esta manera, incorporan innovación en su estrategia corporativa.

En algunas pymes innovadoras los desarrolladores de software libre encuentran trabajo con más facilidad. De hecho, para estas compañías, contratar a estos desarrolladores es una manera de mostrar su implicación en los productos y servicios que ofrecen (cuando están basados en software libre), y una manera de diferenciarse de la competencia. La relación entre pymes y voluntarios en algunos proyectos es tan cercana que los desarrolladores pasan de vez en cuando de una pyme a otra, sin dejar de trabajar en el mismo proyecto y viceversa: cambian de proyecto sin llegar a tener que cambiar de empresa. Desde muchos puntos de vista, muchos proyectos serían simplemente inviables sin la participación de las pymes y los recursos que invierten en el desarrollo, mientras que al mismo tiempo estas pymes no serían viables sin el proyecto exitoso de software libre con el que trabajan.

La relación de las pymes con las grandes corporaciones en las comunidades de desarrollo no es siempre fácil. Las pymes tienen problemas para crear ecosistemas ricos por sí mismas, lo que significa que normalmente agradecen la adhesión de grandes empresas. Pero una vez que entran en juego, puede darse una situación de desequilibrio para los actores más pequeños. Aunque por norma suelen ser más activos y dinámicos, el mero volumen de grandes actores y la disponibilidad de sus recursos puede dirigir el proyecto en direcciones inesperadas. En cualquier caso, cuando estas capacidades diferentes se utilizan con el ánimo de buscar sinergias se puede conseguir un enorme impulso para el proyecto y beneficio para todos los actores.

5. El software libre y los profesionales informáticos

Los profesionales de las TIC se sitúan en el corazón del desarrollo de software libre. A pesar de lo que todavía digan fuentes desinformadas, la mayoría de proyectos de software libre son liderados por profesionales experimentados en el desarrollo de software, normalmente con muchos años de experiencia. En algunos casos, han estado involucrados en el software libre desde siempre y en otros vienen de la industria del software tradicional. Puede que empezaran trabajando en proyectos de software libre como *hobby* en su tiempo libre o puede que lo hicieran debido a una decisión de su compañía. Pero, en cualquier caso, ellos como individuos son el pegamento que mantiene unidos los elementos de los proyectos.

Como hemos mencionado repetidamente en este texto, en muchos casos las empresas están involucradas en proyectos, pero no son "ciudadanos de primera" en ellos. La mayoría de las comunidades de software libre están basadas en las relaciones entre personas, no en acuerdos corporativos. Es por esto que normalmente, el papel de los profesionales informáticos que participan en estas comunidades sea mucho más importante que el de las empresas para las que trabajan. De algún modo, la mayoría de los proyectos de software libre están en realidad liderados por profesionales de las TIC que son reconocidos en la medida en que meriten por ello y se esfuercen en el proyecto.

De hecho, este reconocimiento del mérito y el esfuerzo nos lleva a un nuevo papel para los desarrolladores profesionales. En lugar de estar restringidos por acuerdos privados y reglas empresariales que les impiden mostrar a sus colegas sus capacidades y logros, el juego en el mundo del software libre es exactamente el contrario. Todo el trabajo es público, y todas las acciones son trazables. Los profesionales que hacen un buen trabajo son reconocidos por sus colegas y se llevan su reputación consigo según se mueven de compañía en compañía.

Desde el punto de vista del desarrollo profesional a largo plazo, tan importante para los profesionales de las TIC, el software libre es un mundo de posibilidades. Al poder leer el código, entender nuevas arquitecturas de programas libres, nuevas tecnologías y nuevos paradigmas resultan tareas más sencillas. En conclusión, el software libre brinda nuevas oportunidades para profesionales. Por supuesto que existen algunos riesgos evidentes, como los problemas derivados de exponer públicamente prácticas de desarrollo deficientes, por ejemplo. Pero en general, los profesionales tienen mucho más de lo que beneficiarse cuando trabajan con software libre.

6. Conclusión

El software libre es un nuevo campo en el panorama de las TIC, no porque sea una nueva tecnología sino porque brinda nuevos procesos y posibilidades. Tanto las empresas como los profesionales pueden obtener beneficios si encuentran la manera de aprovechar sus capacidades y pueden alinear sus objetivos con los de la comunidad del software libre. De todos modos, como todo nuevo mundo, es crucial entender las nuevas reglas y explorar las oportunidades emergentes para beneficiarse de él. Por supuesto también existen riesgos, especialmente para aquellos que no se den cuenta de que, en muchos aspectos, el entorno ha cambiado.

Notas

[1] Free Software Foundation < http://www.fsf.org/>.
[2] Open Source Initiative <http://www.opensource.org/>.
[3] OW2 Consortium <http://www.ow2.org/>.
[4] Morfeo Project <http://morfeo-project.org/>.
[5] LiMo Foundation <http://www.limofoundation.org/>.
[6] open handset alliance <http://www.openhandsetalliance.com/>.
[7] Symbian Foundation <http://www.symbian.org/about/index.php>.
[8] GENIVI Alliance <http://www.genivi.org/>.
[9] redhat <http://www.redhat.com/>.
[10] fedora <http://fedoraproject.org/>.
[11] CANONICAL <http://www.canonical.com/>.
[12] ubuntu <http://www.ubuntu.com/>.
[13] NOKIA <http://www.nokia.com/>.
[14] maemo.org <http://maemo.org>.
[15] The Apache Software Foundation <http://www.apache.org/>.
[16] GNOME: The Free Software Desktop Project <http://www.gnome.org/>.
[17] Sun microsystems <http://www.sun.com/>.
[18] Novell <http://www.novell.com/>.
[19] <http://en.wikipedia.org/wiki/Eazel>.
[20] *<http://en.wikipedia.org/wiki/Ximian>*.
[21] mozilla.org <http://www.mozilla.org/>.
[22] Google <http://www.google.com/intl/en/corporate/>.
[23] MySQL <http://www.mysql.com/>.

Buenas prácticas para la adopción del software libre

Carlo Daffara
Director de I+D de Conecta Research ltd., Italia

<cdaffara@conecta.it>

Traducción: Juan Jose Amor Iglesias (GSyC/LibreSoft, Universidad Rey Juan Carlos)

Resumen: *muchas empresas y administraciones públicas se plantean la adopción del software libre (Free/Libre and Open Source Software, FLOSS) en sus proyectos, o simplemente la migración a este, puesto que les puede implicar menores costes de implementación de nuevos sistemas informáticos o de mantenimiento de los actuales. Pero, en general, los procesos de adopción o migración son complejos: se requieren esfuerzos multidisciplinares que cubran diversas áreas, una completa comprensión de cómo son los flujos de trabajo internos en la empresa y de cómo interactúan los usuarios con los sistemas informáticos. Este artículo propone una serie de pautas con el fin de maximizar el éxito de una adopción o migración a software libre en organizaciones, pautas que se dividirán en tres grupos: pautas de gestión (orientadas a la gestión de mayor nivel en la compañía), pautas técnicas (orientadas al punto de vista técnico) y pautas sociales (orientadas a características típicas del FLOSS que las compañías querrán aprovechar, como la relación con las comunidades de desarrollo o la mejora del soporte técnico gracias a la experimentación y formación internas).*

Palabras clave: *adopción de software libre, comunidades de software libre, empresas, migración, software libre.*

Autor

Carlo Daffara es el miembro italiano del Grupo de Trabajo Europeo sobre software libre. Preside además diversos grupos de trabajo como el grupo de *middleware* de código abierto del comité técnico en computación escalable del IEEE y el grupo de trabajo de las PYME del Competitiveness Task Force de la UE. Su actividad investigadora actual se centra en la sostenibilidad de los modelos de negocio basados en el software libre. Actualmente es jefe de I+D en Conecta Research ltd, empresa de consultoría en software libre.

1. Introducción

En una empresa, la decisión de adoptar soluciones basadas en software libre (*Free/Libre Open Source Software*, FLOSS), o la de integrar el software libre en sus servicios, se basa normalmente en los siguientes casos de uso:

- Migración/Sustitución básica: consiste en usar el FLOSS en la infraestructura informática, sustituyendo normalmente instalaciones basadas en software privativo.
- Nuevo desarrollo: introducción del FLOSS para desarrollar un nuevo proyecto de la compañía (adopción).
- Venta de servicios basados en FLOSS.
- Venta de productos que contienen FLOSS internamente en una cantidad significativa.

En este sentido, una empresa puede encontrar útil el software libre desde un punto de vista táctico (cuesta menos dinero implementarlo, con menos dependencia de proveedores, o puede ayudar a introducir productos en el mercado en menos tiempo). Los procesos de adopción interna se suelen modelizar usando un modelo incremental, desarrollado inicialmente por P. Carbone [1] y otros. Este modelo es un proceso de pasos sucesivos, siendo el primero el más complejo de todos, normalmente (ver **figura 1**).

En todo caso, el proceso de migración y adopción es complejo, con esfuerzos multidisciplinares que afectan a varias áreas y requieren una comprensión completa de los flujos de trabajo individuales, cómo se ejecutan y cómo la gente interactúa con los sistemas de información en su trabajo diario. En este sentido, una migración a software libre es un reto importante y, como muchas otras tareas complejas, puede ir mal. Hay muchos obstáculos que nos encontramos al enfrentarnos a una migración, pero algunos pueden evitarse fácilmente siguiendo prácticas simples. Muchas de las dificultades no son de naturaleza técnica, sino organizativas, y casi todo el esfuerzo que requerirán vendrá de arriba (del equipo de gestión de la empresa); otro aspecto importante es el impacto social de la migración (tal como la aceptación de los usuarios de los nuevos sistemas), lo cual requerirá atención aparte.

2. Pautas de gestión

El camino a seguir para hacer una migración correcta al uso de FLOSS siempre empieza por tener un panorama preciso de las funciones de Tecnologías de la Información (TI) de la empresa, una clara visión de las necesidades y beneficios de las transiciones y un soporte permanente. Las diferencias con los modelos de desarrollo y soporte en el software libre pueden requerir un cambio significativo en la manera en la que el software y los servicios son gestionados y, en general, un desplazamiento de la responsabilidad desde los proveedores externos hacia el personal interno.

Asegúrese de gestionar el compromiso a la transición

El soporte de la gestión y el compromiso se muestran repetidamente como las variables más influyentes en el éxito de los esfuerzos más complejos de gestión de TI, y las migraciones a software libre no son una excepción. Este compromiso debe mantenerse durante un período de tiempo suficiente para cubrir la migración completa. Esto significa que, en las organizaciones cuyos directores de Informática son cambiados con frecuencia, o donde la gestión cambia con cierta periodicidad (por ejemplo, en las Administraciones públicas donde ocurre a menudo), debe haber un proceso para transferir la gestión de la migración. El compromiso debe también extenderse a la financiación (puesto que tanto los cambios como la formación requerirán recursos, tanto económicos como humanos). La mejor forma de garantizar una coordinación constante es asignarle un equipo con experiencia mixta (gestión y técnica) para dar una realimentación continua y una gestión del día a día.

A tener en cuenta: si las únicas personas que trabajan en la migración son personal del departamento de informática, probablemente no tendrán información suficiente sobre la gestión de más alto nivel, ni planes de financiación para continuar la migración tras los primeros pasos.

Prepare una panorámica clara de lo que se espera de la migración o adopción, incluyendo puntos medibles

La transición puede empezarse por varios motivos, incluyendo el mejor control de los costes de informática, independencia de los proveedores, flexibilidad o soporte de los estándares abiertos. Para asegurarse de que la migración está produciendo beneficios de forma efectiva, o está yendo de acuerdo al plan de migración, es fundamental conocer de antemano los indicadores que servirán para evaluar el progreso. Los requisitos deben ser realistas, y, en particular, las expectativas de

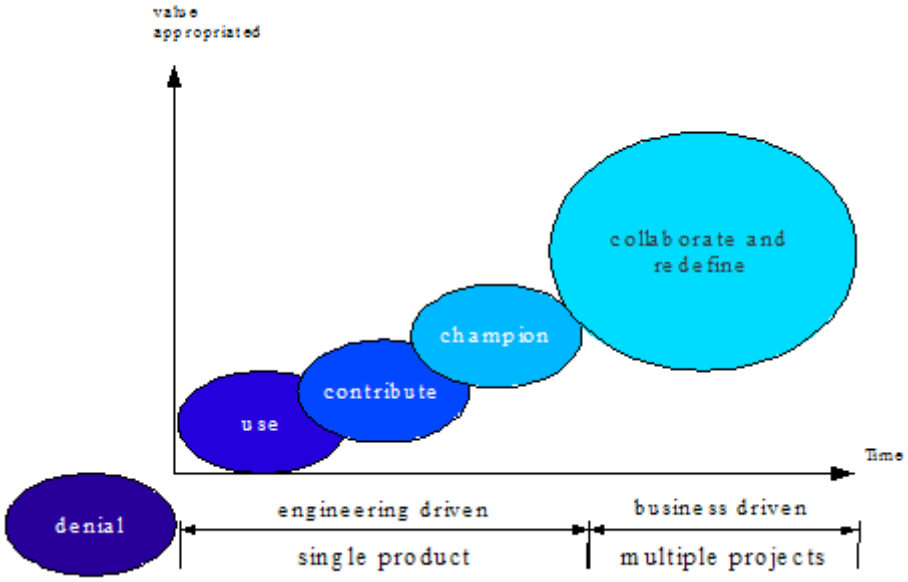

Figura 1. Modelo de pasos sucesivos en la adopción de software libre.

reducción del coste total de la propiedad (TCO) deben poderse contrastar con datos públicamente disponibles.

A tener en cuenta: si la única ventaja que se percibe es que "el software se baja gratis de la red", puede que tengamos una serie de premisas incorrectas que probablemente desembocarán en una opinión negativa sobre la migración.

Asegúrese de que el calendario es realista

La introducción de una nueva plataforma informática requerirá siempre mucho tiempo; por regla general, el tiempo para realizar una transición completa hacia el FLOSS puede considerarse comparable al de la introducción de un nuevo software ERP (Enterprise Resource Planning) en la compañía; para transiciones más pequeñas, el esfuerzo en tiempo debe escalarse proporcionalmente.

A tener en cuenta: cuando el tiempo de migración se mide en días, y no se planifica esfuerzo para después de la migración, el proceso puede quedarse detenido forzosamente si los recursos planificados se agotan.

Revise los procedimientos actuales de obtención y desarrollo de software o tecnología informática

Puesto que los esfuerzos de implementación se desplazan desde el software comercial hacia el libre, los procesos de obtención y desarrollo deben ser actualizados convenientemente. En particular, la mentalidad de "compra" puede que deba cambiarse hacia una mentalidad de "servicio", puesto que se comprará menos software "precintado" y este cambio puede requerir diferencias en el reparto del presupuesto interno para informática. El software desarrollado internamente requerirá ser *portado* o adaptado a un nuevo software que será multi-plataforma o accesible usando interfaces estándar (por ejemplo, aplicaciones web), y esto debe tenerse en cuenta en el plan informático global.

*A tener en cuenta: C*uando no se planifican cambios en el proceso de obtención y desarrollo, es porque no se ha comprendido el alcance de los cambios requeridos para la adopción de software libre en la empresa.

Busque consejo o información de experiencias parecidas

Dado que el número de empresas y administraciones que han hecho una migración ya es considerable, es fácil encontrar información sobre lo que hacer. En este sentido, el proyecto COSPA ha desarrollado una base de conocimiento que es accesible a través de su sitio web principal [2]; las Administraciones públicas pueden también contactar con el centro de competencias en software libre que tengan más a mano, para que les proporcione información y apoyo durante el proceso de migración.

Evite una transición "de golpe" e intente que las migraciones sean incrementales

Casi todas las migraciones a gran escala han sido realizadas de una vez[1] (implicando un cambio abrupto entre un entorno informático y el nuevo) y suelen venir afectadas por costes técnicos y de soporte extremadamente altos. Aunque la necesidad de apoyo técnico en más de un entorno a la vez incrementa tanto los costes de soporte como de gestión, las migraciones incrementales o "suaves" suelen traer una mejor experiencia a los usuarios, así como un trastorno mínimo de los procesos de negocio.

Un ejemplo de migración suave sería aquella que comienza por la migración de las aplicaciones del lado del servidor, que normalmente siguen estándares o simplemente basadas en conceptos de red (cliente/servidor) que las hace fáciles de reemplazar, dejando las aplicaciones de escritorio o de interfaz de usuario para el final. Un esquema así se mostró en [3] (Ver **figuras 2 y 3**).

Asigne al menos una persona que interactúe con la comunidad de desarrollo e intente encontrar buenas fuentes de información en la red

Una ventaja importante del software libre es la disponibilidad en la Red de recursos gratuitos, en forma de bases de conocimiento, listas de correo, wikis (sitios cooperativos) que pueden proporcionar una cantidad de apoyo técnico que, en muchos casos, es comparable a lo que ofrece un servicio comercial.

El gran problema es la localización de esas fuentes de conocimiento; en este sentido, asignar recursos para encontrar, clasificar e interactuar con esas fuentes de información es una forma de reducir el coste del soporte; una manera de ofrecer de forma unificada estas fuentes de información es mediante una sencilla página en la Intranet con enlaces a recursos públicos en línea.

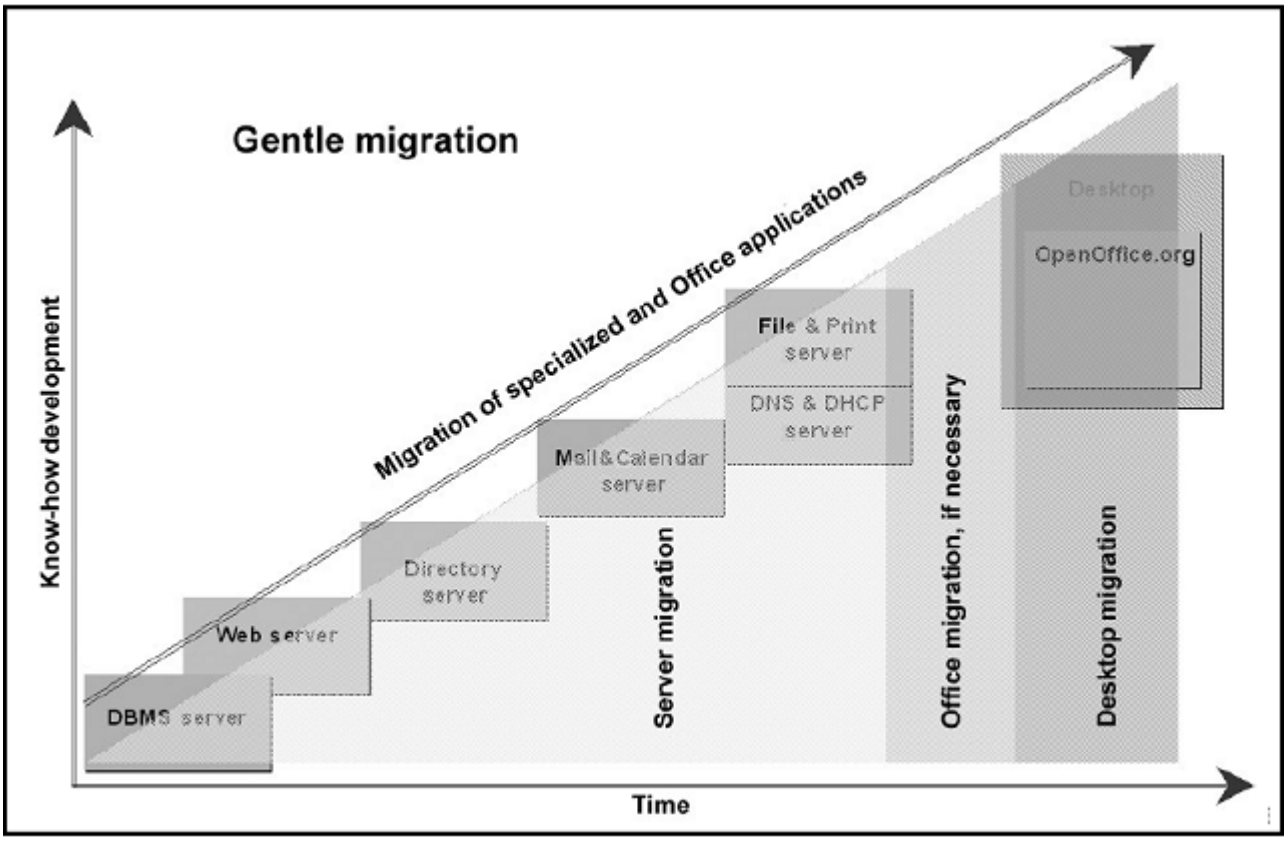

Figura 2. Esquema de migración "suave»"comenzando por las aplicaciones del servidor.

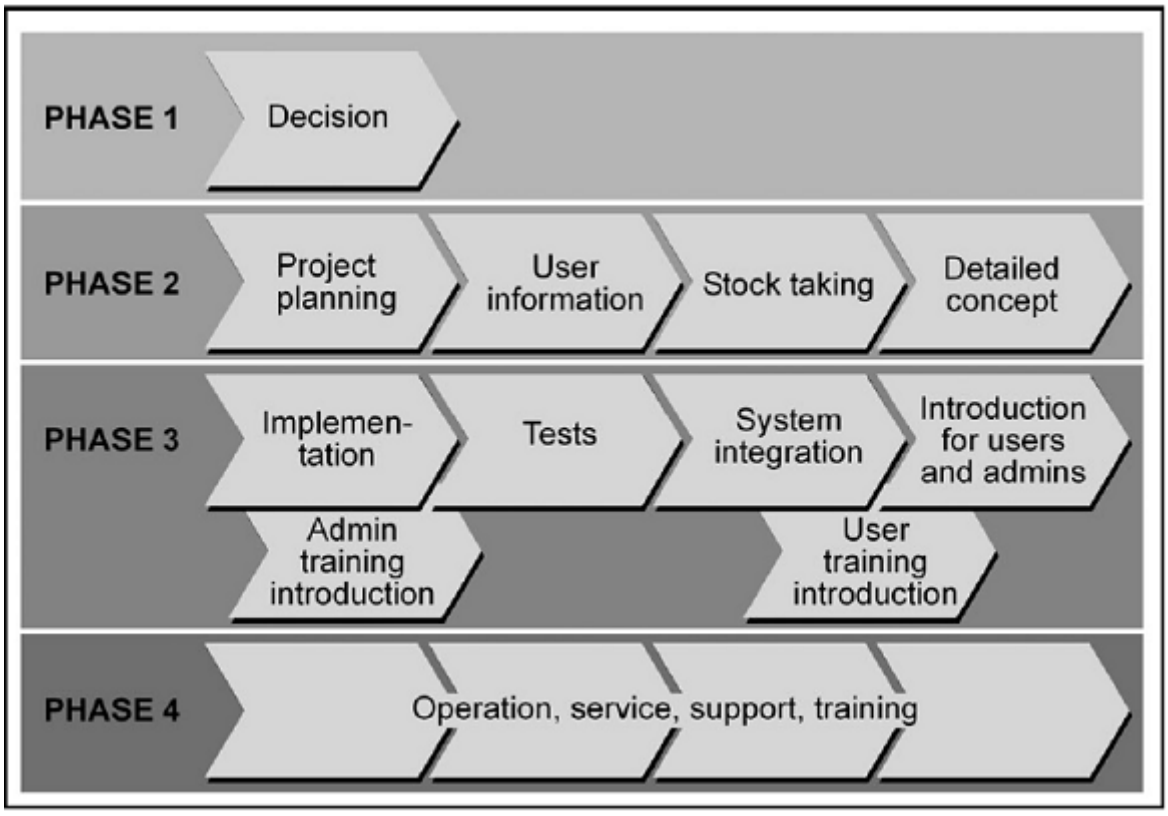

Figura 3. Esquema de fases de una migración "suave".

A tener en cuenta: vigilar si la gente no sabe cómo encontrar información sobre las herramientas que utilizan, o cuándo todos tienen que buscar en los sitios web para encontrar ayuda.

3. Pautas técnicas

Una diferencia que caracteriza las adopciones de FLOSS es el modelo de desarrollo adoptado por casi todos los proyectos de software libre, que afecta especialmente a la frecuencia de entregas de actualizaciones o soporte. Esto requiere un cambio en la manera de adoptar el FLOSS y gestionar las actualizaciones, para reducir en lo posible los problemas de interoperabilidad.

Entienda la manera de desarrollar el software libre

Casi todos los proyectos siguen un modelo de desarrollo cooperativo, con un grupo central de programadores haciendo casi todo el código (normalmente desde una empresa) y un gran número de cooperadores adicionales. Este modelo proporciona una gran calidad en el código y rápidos ciclos de desarrollo, pero también requiere un esfuerzo importante en el seguimiento de los cambios y las actualizaciones. La adopción de un paquete de software libre debe considerarse cuando:

- El proyecto en sí mismo está "vivo", es decir, tiene alrededor una comunidad de desarrollo activa.
- Hay una clara distinción entre software "estable" e "inestable". En muchos proyectos, hay dos líneas de desarrollo: una, centrada en integrar los últimos cambios e implementaciones (la "inestable"), y otra, centrada en mejorar la estabilidad y corregir los fallos (la "estable"); periódicamente, los programadores "congelarán" el desarrollo para convertir una versión "inestable" en otra "estable", creando aparte una nueva rama de desarrollo con las últimas novedades implementadas. Esta distinción permite a los programadores satisfacer tanto a los usuarios que quieren probar las últimas funcionalidades, como a aquellos que usan el software en sistemas de producción, pero requiere un esfuerzo extra para recolectar la información sobre las nuevas versiones.

Si las nuevas funcionalidades o correcciones son necesarias, puede ser más simple pedir una versión con soporte comercial; en muchos casos, el proveedor comercial será también un cooperador económico en el proyecto de software libre que vende.

A tener en cuenta: precaución cuando el gestor de TI o los desarrolladores creen que el software libre viene a ser como un software comercial que alguien ha puesto gratis en la red y simplemente "funciona".

Prepare una lista completa de software y hardware que serán afectados por la migración y qué funcionalidad está buscando la compañía

Cuando la situación inicial es poco conocida, la migración puede ser un fracaso. Muchas empresas y Administraciones no tienen implementadas auditorías de las plataformas de software y hardware, y no están capacitadas para cuantificar las herramientas y el software que deben ser reemplazados o integrados en una migración a software libre. El proceso de elaboración de la lista debe tener en cuenta también el número de usuarios simultáneos, el uso medio en la organización y si las aplicaciones utilizan protocolos de comunicación o formatos de intercambio abiertos o cerrados.

Esta encuesta será la base de la decisión sobre a qué usuarios afectará la migración en primer lugar, y para tener en cuenta el coste de la adaptación del software o la migración a un formato de datos diferente. Las herramientas de inventario automático de software están ahí y pueden reducir el coste de realizar el inventario y permitir un control más estricto sobre el software instalado (reduciendo el coste de mantenimiento).

Algunos aspectos a verificar:

- El formato de datos usado, a nivel de intercambio de documentos pero también a nivel de base de datos y protocolos de comunicación de red.
- La lista de aplicaciones en uso, incluyendo aquellas desarrolladas internamente, macros y documentos.
- La funcionalidad disponible.
- Las deficiencias o los problemas de la infraestructura actual.

Es fundamental que el software migrado cubra los mismos requisitos funcionales de la infraestructura informática actual, y es preferible que además mejore la misma en términos de funcionalidad o calidad (disponibilidad, fiabilidad, rendimiento...).

Utilice la flexibilidad del software libre para hacer adaptaciones locales

Lo que hace diferente al software libre es la flexibilidad y la libertad que da a los usuarios y programadores para crear nuevas versiones o adaptaciones de cualquier paquete. Esta flexibilidad puede aumentar mucho el valor que se percibe del software libre, por ejemplo es posible crear paquetes personalizados que contienen configuraciones locales, tipos de letra específicos u otro material de interés como macros prediseñadas o plantillas que se usen habitualmente en la empresa. Además, la personalización del aspecto visual puede aumentar la aceptación de los usuarios, al presentarles un escritorio más atractivo que mantenga a mano los enlaces o elementos de menú más habituales.

Esta personalización puede integrarse de forma simple en casi cualquier distribución de Linux, o bien creando un repositorio de software local. Obsérvese que, en casi todos los casos, no hay que escribir programas, puesto que la personalización consiste en seleccionar el paquete adecuado, cambiar el aspecto visual o proporcionar plantillas de documentos.

Hay mucho más software disponible que el que se instala por defecto

Temas de licencia o el propio diseño son los que limitan sustancialmente la cantidad de software que normalmente se incluye en las instalaciones predefinidas de las distribuciones Linux habituales. Por ejemplo, solo unas pocas incluyen capacidad para reproducir los formatos de vídeo o audio más populares, debido a restricciones de licencia o patente; también es habitual que algunos paquetes que nos parezcan interesantes no estén incluidos porque solo los usan una minoría de los usuarios.

Por esta razón, es importante investigar e incluir en las instalaciones predefinidas aquellos paquetes adicionales que puedan ayudar en el período de transición, como paquetes que incluyan tipos de letra adicionales, herramientas multimedia y otros paquetes que puedan ser útiles en un entorno mixto.

Al seleccionar los paquetes, dé preferencia siempre a la estabilidad sobre la funcionalidad. Entre los muchos paquetes potencialmente útiles para cada función, siempre los hay con más funcionalidades o con más estabilidad. En general, entre los paquetes candidatos deben elegirse los más estables, o aquellos que sean más populares (pues esto generalmente hace que haya más información de soporte disponible), y también aquellos con menor variación entre versiones.

A tener en cuenta: Cuidado si el administrador de TI siempre quiere poner lo último en los escritorios de los usuarios.

Diseñe la infraestructura de soporte para reducir el número de incompatibilidades

Cada transición de una infraestructura informática a otra siempre conlleva pequeñas diferencias e incompatibilidades; esto puede verse por ejemplo al convertir documentos de un formato a otro.

La infraestructura general debe reducir el número de estos pequeños problemas, por ejemplo rediseñando las plantillas de documento en el formato abierto ODT (*OpenDocument Text*) en lugar de usar versiones previas hechas con herramientas privativas. Esto reduce en gran parte las diferencias de estilo y formato que se producen al convertir un documento de un formato a otro.

Introduzca un sistema de gestión de incidencias[2]

Una dificultad en cada nuevo despliegue tecnológico es cómo asegurarse la satisfacción de los usuarios y el grado de aceptación de la nueva solución, especialmente en compañías de tamaño medio donde es más difícil obtener realimentación de los usuarios. Un sistema de gestión de incidencias puede ofrecer una forma simple de recolectar los fallos del despliegue y ayudar a identificar si los usuarios necesitan capacitación adicional, por medio del análisis de las estadísticas de envío de incidencias por usuario. Además puede encontrar puntos débiles del despliegue, por ejemplo cuando se localizan muchas incidencias relacionadas con un área concreta.

Recopile los detalles de la migración en un diario

Una migración a gran escala requiere una acción coordinada e información clara y actualizada. La mejor forma de recopilar esta información es manteniendo un "diario de migración", un lugar donde tener la documentación recolectada y preparada para la migración (incluyendo los objetivos, el plan detallado y la documentación técnica), y un diagrama de tiempos actualizado convenientemente de acuerdo con el avance del proyecto. Esto además simplifica la gestión del proyecto cuando hay un cambio en el equipo que dirige la migración.

4. Pautas sociales

Proporcione información general sobre software libre

Un obstáculo habitual en la adopción de software libre es la aceptación de los usuarios, quienes suelen tener pocas nociones sobre software libre y estándares abiertos. Casi siempre, el software libre se percibe como peor, dado que es "gratis", al poderse obtener fácilmente de la red como muchos paquetes *shareware* o hechos por aficionados. Es importante acabar con esta percepción y proporcionar información sobre cómo se desarrolla el software libre y qué ideas generales y de negocio subyacen en él.

No fuerce a los usuarios a cambiar, déles explicaciones

El cambio en la infraestructura tecnológica provocará un cambio importante en la forma de trabajar de los usuarios o en el uso de los recursos internos; este cambio puede causar resistencia en esos usuarios. Por ello debe simplificarse el cambio, explicando claramente el por qué y cómo tendrá lugar y qué beneficios se obtendrán a largo plazo, tanto internamente (menor coste, mayor flexibilidad y seguridad) como externamente (apertura, cumplimiento de estándares internacionales, menor carga a los usuarios externos).

A tener en cuenta: Hay que tener cuidado cuando los usuarios creen que la migración simplemente se hace para ahorrar dinero en el software.

Aproveche la migración como una manera de mejorar las habilidades del personal

Puesto que la nueva infraestructura requerirá capacitación, se puede aprovechar para mejorar las habilidades generales sobre TI. En muchas empresas y Administraciones públicas, se imparten pequeños cursos formales a los usuarios. Esto no solo incrementa la confianza, sino que armoniza habilidades en el grupo y en general mejora el desempeño.

Esto puede provocar cierta resistencia por parte de los "gurús de la empresa", que pueden interpretar esta formación como un mecanismo para destronarles como líderes técnicos. La mejor forma de evitar esta resistencia es localizar a esos usuarios y sugerirles que repasen el material de formación de mayor nivel (que puede estar disponible en un sitio web, por ejemplo).

Además, puede ser interesante localizar a los "campeones", es decir, entusiastas del software libre que pueden ayudar dando soporte a otros usuarios y ofrecerles formación adicional o un reconocimiento a nivel de gestión.

En general, es útil crear una Intranet interna con enlaces a los diferentes paquetes de formación.

Facilite la experimentación y el aprendizaje

La libertad de licenciamiento, fundamental en el movimiento del software libre, permite la libre redistribución del software y los materiales de formación. En este sentido, entregar a los usuarios distribuciones en CD "live" (que no requieren instalación en el disco duro para probarlas) o material impreso que puedan llevarse a casa, puede aumentar la aceptación general del nuevo software.

Agradecimientos

Este artículo se basa en material preparado para los proyectos COSPA, OpenTTT y FLOSSMETRICS financiados por la UE. Puede conseguirse más información en la guía FLOSSMETRICS para PyMES (FLOSSMETRICS guide for SMEs), <http://guide.conecta.it>.

Referencias

[1] P. Carbone. *Value Derived from Open Source is a Function of Maturity Levels*, OCRI conference "Alchemy of open source businesses", 2007. <http://www.ocri.ca/email_broadcasts/041907 partnership.html>.

[2] EU COSPA Project. *D6.1 Report evaluating the costs/benefits of a transition towards ODS/OS.* <http://www.cospa-project.org/>.

[3] KBSt, Alemania. *Migration guide, 2006.*

Notas

[1] Esto enfoque es conocido también como "enfoque Big Bang" o "adopción Big Bang" <http://en.wikipedia.org/wiki/Product_Software_Adoption:_Big_Bang_Adoption>.

[2] Un caso típico de estos sistemas son los llamados "Trouble Ticket Systems" (o también "Issue Tracking Systems") en los cuales se introducen las incidencias como "tickets", que son asignados a personas o equipos de trabajo que se encargan de resolverlos <http://en.wikipedia.org/wiki/Issue_tracking_system>.

Construir y mantener una comunidad de práctica: método aplicado a proyectos de software libre

Stéphane Ribas, Michel Cezon
Capítulo Local de OW2 Europe, Saint Ismier Cedex (Francia)

<{stephane.ribas,michel.cezon}@inria.fr>

Traducción: Daniel Izquierdo Cortázar (GSyC/LibreSoft, Universidad Rey Juan Carlos)

Resumen: *tecnologías como la Web 2.0 han fomentado el desarrollo de nuevas formas de colaboración en el dominio I+D, evolucionando a menudo hacia la creación de comunidades organizadas. El incremento de las conexiones a nivel internacional ha sacado a la luz la necesidad de colaboración remota alrededor del mundo, lo cual enfatiza la necesidad de organizar la creación y mantenimiento de dichas comunidades. Este artículo presenta un método, desarrollado por el Instituto INRIA, para construir y mantener una comunidad de práctica. Nuestro enfoque, basado en cinco pasos, ha sido evaluado en un proyecto europeo: el proyecto ASPIRE-RFID. Los resultados son aquí revisados, con la convicción de que la transparencia, la confianza y el compromiso de las personas son la clave para allanar el camino hacia el éxito. Crear una comunidad desde cero y desarrollarla satisfactoriamente no siempre surge espontáneamente sino que ha de basarse en un enfoque estructurado. Presentando este método esperamos fomentar el que cualquier persona que desee crear una comunidad se lance a la aventura de un modo más productivo.*

Palabras clave: *compartir conocimiento, comunidades, entornos colaborativos, metodología, relaciones sociales, software libre, Web 2.0.*

Autores

Stéphane Ribas (M.Sc, University of Surrey 1996) ha trabajado 12 años en la industria del software y servicios, mayormente dedicado a tecnologías innovadoras tales como comercio electrónico, sistemas de salud electrónicos, seguridad en las telecomunicaciones, Intranet, sistemas expertos, *middleware*, aplicaciones de servidor, Java y software libre. Ha trabajado muchos años en varios países europeos y ha participado en diversos e importantes proyectos ofreciendo soporte y consultoría técnica para grandes clientes. Ha desarrollado importantes cualidades en la construcción y promoción de comunidades en Internet. Gracias a esas experiencias, ha ido adquiriendo un bagaje técnico muy sólido en software. Comenzó a trabajar en INRIA en 2008 para coliderar el Capítulo Local de OW2 Europa y contribuir en diversos proyectos de software libre y grupos de trabajo (Xwiki Concerto, AspireRFID, QualiPSo, NESSI OSS Working Group).

Michel Cezon ha trabajado alrededor de 25 años en industrias y servicios informáticos, mayormente en tecnologías innovadoras (inteligencia artificial, NLP, negocios electrónicos, EDMS, Intranet). Lideró numerosos proyectos europeos (FP4 EPTO-1993, FP6 SUPREME-1997, FP6 RENAISSANCE-1997, FP6 PRORAD-1998) además de grandes consorcios de carácter internacional (FP4 MOVIT-1995, United Nations/WIPO-2000), o proyectos multimillonarios para grandes clientes. Graduado como Ingeniero en Informática y especializado en inteligencia artificial, robótica y visión artificial, posee un dominio de la gestión de proyectos y de las habilidades de calidad aprendido en grandes compañías como Hewlett Packard o Cap Gemini Ernst and Young. Siendo consciente de las cuestiones de ámbito multicultural, pasó años en Singapur, Estados Unidos y Suiza, viajando frecuentemente alrededor de Europa. Se unió a INRIA en enero de 2007 para gestionar y coordinar proyectos europeos. Actualmente está trabajando en diversos proyectos europeos financiados por FP6 y FP7 (NESSI-Soft, NESSI-Grid, Qualipso, AspireRFID) y colidera el Capítulo Local de OW2 Europa.

1. Introducción

En el dominio de I+D, tecnologías como Web 2.0 han fomentado el desarrollo de nuevos modos de trabajo colaborativo, a menudo basados en relaciones entre pares (*peer-to-peer*) que finalmente evolucionan hacia comunidades más organizadas.

La creación de dichas comunidades ha mostrado su eficiencia y dinamismo en diversos dominios, más específicamente en el desarrollo de software libre. Y aún más, el incremento de las conexiones a nivel internacional ha generado una nueva necesidad de colaboración remota entre diferentes comunidades de expertos alrededor del mundo.

Desarrollado dentro del instituto INRIA, basado en la experiencia del Capítulo Local de OW2 Europa [1] y en la creación de la comunidad Aspire-RFID [2], este artículo describe una metodología para crear y mantener una comunidad de práctica.

2. ¿Qué es una comunidad?

Una comunidad puede ser considerada como un grupo de personas que comparten un interés común, las mismas inquietudes o la misma pasión por algo. Una comunidad puede estar también basada en roles o especialidades (por ejemplo administrador de redes, grupos de usuarios de Linux o el grupo de usuarios de un sintetizador modular). Estas personas mejoran y profundizan en sus conocimientos cuando los comparten, ya sea resolviendo problemas para otros, interactuando regularmente con los miembros de la comunidad, preguntando y resolviendo preguntas, o reutilizando buenas ideas [3].

Estos entusiastas participan activamente y se crea una nueva identidad virtual basada en un fuerte vínculo social que es de esperar produzca resultados colectivos [4].

3. Tipologías de las comunidades

Brevemente, podemos distinguir cuatro tipos de comunidades [3]: Comunidad de aprendizaje (por ej. el Proyecto Plume [5]), comunidad de interés (por ej. Audiofanzine [18]), comunidad de apasionados (por ej. administradores de sistemas participando en la Conferencia JRES 2009 [19]), y comunidad de práctica (por ej. Linux User Group [20]).

La comunidad de aprendizaje consiste en un grupo de personas que comparten valores y creencias comunes y que aprenden activamente unos de otros. Dichas comunidades han llegado a ser la plantilla de inicio para otros tipos de comunidades. Se basan en un tipo avanzado de diseño educativo o pedagógico considerado a menudo como un enfoque interdisciplinar para la educación superior [6]. Los participantes en comunidades de aprendizaje deben sentir algún tipo de lealtad con el grupo que dirige sus esfuerzos al trabajo continuo y a ayudar a otros, influyendo en lo que ocurre en la comunidad (de manera activa o reactiva). Este tipo de comunidades son lo suficientemente flexibles como para dar la oportunidad a los participantes de expresar opiniones personales, pedir ayuda o información específica y compartir historias de eventos [4][6].

Una comunidad de interés podría ser considerada como un grupo de personas que com-

parten temas que no requieren una comunidad formal sino discusiones más hilvanadas para la colaboración y la compartición de conocimiento. Representan grupos de personas no demasiado unidas entre sí y sin un compromiso claro en términos de resultados. Están bien al corriente de los desarrollos en los temas de su interés y se dedican a formular y responder preguntas [4].

Una comunidad de apasionados está formada por un grupo de gente con un conjunto de actividades, formas de gobernabilidad y estructura más formales. Los miembros tienen un rol en concreto (por ej. "asesor de seguridad de redes"), y ayudan activamente a otros miembros a encajar y evolucionar en su rol, persiguiendo todos ellos obtener un dominio completo de su disciplina [4].

Las comunidades de práctica tienen una estructura menos formal y están basadas en especialidades de trabajo comunes entre sus integrantes. Los miembros tienen un rol particular o una especialidad concreta (por ej. seguridad) enfocado a la adquisición de experiencia y desarrollo de facultades en ese rol o especialidad. Un factor de motivación importante es el aprendizaje sobre la especialidad y la resolución de problemas [4].

La metodología discutida en este artículo se diseño y aplicó a este último tipo de comunidad.

4. Método para construir y mantener comunidades de práctica

La metodología está formada por diversos pasos, comenzando desde un "objetivo común" hasta la "monitorización" (ver **figura 1**). Las diferentes fases pueden ser organizadas en cinco categorías principales [17]: Análisis, construcción, promoción, mantenimiento y monitorización.

4.1. Análisis

El objetivo de la fase de análisis [3][4][7] consiste en comenzar el proceso y recoger toda la información relevante que permita comprender lo que queremos hacer, a dónde queremos ir, por qué y cómo. Como ya hemos mencionado, una comunidad puede emerger desde cero, a través de un idea común de un pequeño grupo de personas entusiastas, pero que para llegar a tener éxito, necesita un poco más de organización y gestión interna.

En esta fase, debes comenzar "soñando" sobre: ¿cómo trabajaría la comunidad idealmente?, ¿cómo debe estar organizada?, ¿quién liderará y tomará decisiones? En definitiva, deja volar tu imaginación libremente.

La identificación de la temática y subtemáticas permitirá aclarar tus ideas y compartirlas con tus colegas. Llegados a este punto, identificar el equipo principal también es importante y conocer quién está comprometido y deseando participar en este viaje.

Una vez que el equipo ha sido definido, debes tratar los temas básicos: concretamente ¿cómo se va a organizar el trabajo en equipo? Esto implica más pensamiento imaginativo y quizás sesiones de tormentas de ideas y de intercambio de información. Todos estos procesos abarcarán el gobierno interno, la comunicación, el ciclo de desarrollo, el entorno colaborativo, diseminación y promoción. De nuevo papel y lápiz deberán ser reemplazados por algunas herramientas más sofisticadas (preferiblemente software libre) para permitir un mejor y más eficiente trabajo en grupo. Identifica (y usa) las herramientas de la comunidad para organizar tus colaboraciones, tu contenido y facilitar tus actividades de diseminación (promoción y concienciación). Estas herramientas estarán centradas alrededor una forja de software que permitirá una interacción y comunicación diarias, siendo el repositorio central de la comunidad y del proyecto. La metodología permite usar un conjunto de herramientas que ayuden durante esta fase.

A través de este proceso, debemos llegar a una idea mejor acerca de qué es lo que intentamos construir y llevar a cabo. Podría ser deseable parar en cierto punto y echar un vistazo alrededor para ver si ya existen comunidades o proyectos que compartan las mismas inquietudes y temas. Unirse a dichas iniciativas o comenzar desde cero requiere una decisión bien meditada y conlleva dos hojas de ruta diferentes.

Al final de la fase de análisis debe tomarse una decisión tipo "*go-no-go*", es decir, qué hay que hacer y qué no, en base a toda la información recolectada y a la hoja de ruta planificada. Los criterios para tomar tal decisión pueden ser objeto de un nuevo artículo en el futuro.

Una decisión de tipo "*no-go*" necesitará ser explicada en detalle y cuidadosamente para identificar cuales son los mayores impedimentos, dado el actual contexto, y evitar así realizar de nuevo la misma pregunta una y otra vez, cuando la iniciativa o las ganas de trabajar aparezcan nuevamente sobre la mesa.

Una decisión '*go*' es incluso más difícil porque el tiempo y el esfuerzo gastado hasta el momento son más pequeños de lo que realmente será necesario para iniciar y construir la comunidad. Un compromiso fuerte y una clara comprensión del camino a seguir y de los esfuerzos necesarios son obligatorios y deben ser conocidos por todos los miembros principales del equipo.

Si seleccionaste la opción de comenzar una comunidad desde cero, debes pasar ahora a la segunda fase principal: la construcción.

4.2. Construcción

En esta fase es donde pones toda la carne en el asador. Comienzas codificando, creando documentación (por ejemplo guías de usuario, APIs, guías de desarrollo, casos de uso, etc.), configuras tu portal (lema de tu proyecto, licencia de uso, estado de desarrollo, cómo instalar el software, sección de descargas, capturas de pantalla, etc.), publicas paquetes de software, publicas tareas del proyecto y su planificación, gestionas voluntarios y contribuciones externas [8]. Tus actividades diarias serán guiadas por el gobierno de la comunidad, política de propiedad intelectual, ciclo de vida del proyecto y reglas de comunicación (simples y flexibles). Detallaremos más información sobre esta fase en un artículo futuro.

Una vez completadas estas tareas, es hora de darte a conocer.

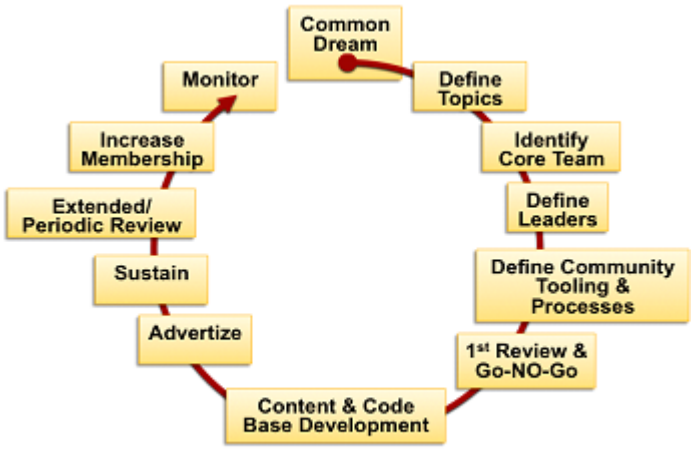

Figura 1. Método para crear y mantener una comunidad.

4.3. Promoción

Tienes un equipo trabajando, un código fuente base, documentación y un portal donde la gente puede encontrar todo lo que necesitan saber, ya sea para contribuir o para usar tu código fuente. ¡Básicamente algo que mostrar! Tu comunidad se ha establecido, así que ahora debes publicitarla: envía artículos a listas de noticias, usa listas de correo y envía mensajes a múltiples personas, usa tu propia red de contactos a tu alrededor y anima a los miembros de tu comunidad a hacer lo mismo. Pueden publicar noticias en periódicos locales e internacionales y revistas *online*. Podrías también usar conocidas redes sociales (por ej. LinkedIn) para dar a conocer tu comunidad. Una vez que tu comunidad está a pleno funcionamiento y has llegado a cierto nivel de reconocimiento, podrías acceder a la última fase: Mantenimiento.

4.4. Mantenimiento

Necesitas seguir en la cresta de la ola y mantener ese impulso conseguido y, como consecuencia, los líderes de la comunidad deberían dedicar tiempo y liderazgo a animar y estimular la comunidad [14][16]. Recomendamos que lleves a cabo eventos periódicos (por ej. jornadas informativas y de desarrollo, asambleas generales), audioconferencias y publicación regular de resultados en forma de notas, ser activo en foros, y algunas veces incluso copiar temas interesantes que provengan de fuentes de datos externas en tus noticias y foros internos. Recomendamos que seas transparente en cualquier decisión y que además escuches a la gente (permite que sea posible discutir cualquier tema o problema) [9][11].

También necesitas alcanzar una masa crítica de miembros [15] que lleven a cabo tus operaciones diarias (administración de sistemas, gestores de errores, correctores de errores, webmaster, gestor del proyecto, traductores...). Por tanto, tú también necesitas incrementar el número de personas en tu comunidad: propón iniciativas (por ej. concursos de codificación, *hackmeeting*, encuentros entre desarrolladores y usuarios, reuniones de proyecto, etc.), usa las redes sociales, explica cómo la gente puede participar y llegar a ser miembro u ofrece incentivos para unirse. No dudes en cruzar los límites de tu comunidad y recoger buenas ideas de otras comunidades (por ej. comunidades de video-jugadores *online*) [12].

Otros dos factores que son muy importantes para los usuarios externos que vayan a contribuir activamente en una comunidad son: el reconocimiento [13] y la compartición de conocimiento. Podrías proponer que los últimos usuarios registrados o los desarrolladores que más errores arreglan aparezcan en la página web de inicio. Después, debes dedicar recursos a organizar el conocimiento dentro de la comunidad: propón que haya maneras de compartir ese conocimiento (por ej. seminarios *online* [10], talleres, casos de uso, etc.).

Tal como dijimos en la fase de análisis, debes echar un vistazo a tu alrededor y buscar cualquier comunidad o proyecto que pueda estar interesado en usar tu código fuente o que pueda estar interesado en cooperar contigo. Identifica esas comunidades, acércate a sus líderes y proponles colaboración [3].

4.5. Monitorización

Para comprobar el progreso, las tendencias y validar los resultados de tus acciones, te recomendamos que definas tu propio sistema de monitorización mejor que usar plantillas predefinidas: identifica el objetivo (¿por qué y qué querrías medir?), define las métricas, recoge los datos (analiza), identifica problemas y tendencias, define consecuentemente las acciones necesarias para hacer frente a ambos, ejecuta esas acciones y valida una vez más la salud del proyecto.

Te recomendamos que monitorices el crecimiento de tu comunidad, pero también la salud de tu proyecto, usando para ello métricas tales como número de errores generados, número de descargas, número de visitas, número de mensajes en los foros e intercambios de correos, número de eventos organizados, etc. Finalmente, ten en cuenta que debes monitorizar también la comunidad globalmente [9]: ¿Estás en una fase inicial de tu comunidad? ¿O en una fase de crecimiento de comunidad? ¿O en una fase de madurez? ¿O estás en el punto donde la comunidad está declinando (o reviviendo)? Una encuesta podría ayudarte a descubrir en qué punto se encuentra tu comunidad de desarrollo. Ofreceremos más pautas de actuación en un artículo futuro.

5. La comunidad del proyecto AspireRFID

Se trata de un proyecto parcialmente financiado por la Comisión Europea en su programa de I+D FP7. El proyecto ASPIRE se propone el desarrollo y la promoción de *middleware* libre, ligero, que siga los estándares, escalable, respetuoso con la privacidad e integrado, enfocado al desarrollo, el despliegue y la gestión de aplicaciones basadas en RFID y en sensores.

El consorcio ASPIRE quiso crear una comunidad vibrante alrededor de su marco de trabajo RFID basado en software libre y así asegurar el tiempo de vida del código base más allá del periodo de financiación del proyecto europeo. Por eso surgió la idea de crear una comunidad de práctica (y pasión) alrededor del código base.

Usamos este proyecto como una manera de comprobar que nuestro método era válido a través de la auditoría del proyecto y diversas recomendaciones. Al mismo tiempo, validamos la metodología y afinamos la primera versión de configuración y herramientas.

Estando implicados en el desarrollo de ambos proyectos, ASPIRE y la comunidad OW2, encontramos rápidamente un punto de encuentro común en el dominio RFID: por una parte, el proyecto ASPIRE buscaba el desarrollo de un marco de trabajo basado en software libre para RFID y por otra parte, la iniciativa RFID fue promocionada por el consorcio OW2. Lo primero de todo consistió en despegar, mientras que el segundo paso consistió en buscar nuevas contribuciones. Fue una manera perfecta de trabajar con un gran beneficio mutuo.

Hasta ahora debemos resaltar tres objetivos cumplidos:

Incrementar la velocidad de la fase de inicio: mediante la metodología, hicimos una revisión de la comunidad existente y descubrimos que el consorcio OW2 podría ayudar en la creación de la comunidad AspireRFID: ellos tienen unas herramientas excelentes de comunidad que encajan con las necesidades, en cuanto a creación de comunidad, de AspireRFID. OW2 propuso una forja de tal manera que los desarrolladores pudieran ponerse a implementar rápidamente. Además, OW2 propuso también herramientas de difusión y promoción.

Mejorar la diseminación y la promoción: aplicamos el método y conseguimos resultados rápidamente, gracias a la promoción con acciones concretas como artículos, reuniones de desarrolladores, *hackmeetings*, conferencias de desarrolladores, etc. Encontramos que la lista de correo de OW2 fue un gran canal de revalorización y promoción.

Finalmente, el éxito de la diseminación y promoción incrementó los resultados de explotación: en un estudio comparativo sobre RFID realizado por el consorcio industrial PICOM [21] el marco de trabajo de ASPIRE fue puesto a prueba con productos comerciales. A pesar del hecho de que el proyecto no estaba finalizado y el marco estaba aún en desarrollo, ASPIRE cosechó mejores resultados que sus competidores. Podríamos suponer que el método ayudó a reforzar la valoración del producto, a publicitar su hoja de ruta, a mejorar su reputación y a asegurar la vida del proyecto a largo plazo.

6. Conclusión

Como en todo nuevo proceso, el método necesita ajustes y mejoras. Estamos trabajando para integrar nuestros descubrimientos en una versión revisada, pero a continuación exponemos los resultados más importantes:

- El método debería ser aplicado de manera

correcta desde el principio para maximizar los beneficios; el proyecto comenzó unos meses antes de que esta metodología fuera aplicada.

- El compromiso de las personas es clave. La comunidad está basada en personas y no hay métodos o herramientas que puedan reemplazar la buena voluntad y disponibilidad.
- Dedicar tiempo a la transmisión de conocimiento es importante, así como encontrar a personal con cierto nivel de creatividad y liderazgo.
- La visión de la comunidad debe ser compartida entre los miembros y claramente publicitada.
- Los pasos de la metodología no son secuenciales; el solapamiento de procesos impulsará el calendario de trabajo.
- El proyecto AspireRFID es un proyecto del FP7 donde el consorcio tiene que llevar a cabo ciertos entregables basados en una hoja de ruta que puede diferir de los deseos de los miembros de la comunidad. Los conflictos de intereses deben ser identificados y tratados públicamente tan pronto como sea posible.

Definitivamente, el método podría ser mejorado: la siguiente versión está en camino. Lanzar una comunidad y desarrollarla satisfactoriamente no es algo que surja espontáneamente (y por suerte), sino que se basa en un enfoque estructurado que se podría beneficiar de una metodología como la presentada en este artículo.

7. El consorcio OW2 y el capítulo local europeo

OW2 es una comunidad de software libre de carácter global, cuyo objetivo es el desarrollo de *middleware* distribuido bajo licencias libres, presentado en forma de componentes flexibles y adaptables. Estos componentes van desde marcos específicos de software y protocolos hasta plataformas integradas. Los desarrollos de OW2 siguen una aproximación basada en componentes. El consorcio es una organización independiente sin ánimo de lucro abierta a empresas, administraciones públicas, universidades e individuos. La misión de OW2 consiste en desarrollar software libre *middleware* e impulsar una comunidad vibrante y un ecosistema de negocios.

Un Capítulo Local es un grupo de contribuidores que desean unir sus esfuerzos y promocionar los objetivos de un consorcio dentro de una comunidad caracterizada por su geografía o su lengua. Para conseguir representar a OW2 a nivel europeo, el capítulo local europeo fue aprobado y lanzado por el Consejo de OW2 el 15 de mayo de 2008. El capítulo local europeo generó una primera versión de sus estatutos y ha definido su estrategia través de flujos de trabajo y tareas: impulsar la comunidad de investigación y académica de OW2 en Europa, ayudar en el desarrollo del ecosistema de negocios de OW2 en Europa y proporcionar apoyo específico a las necesidades de la comunidad local.

Los objetivos se abordan a través de tres grandes líneas de acción: Comunicación con la parte académica, concienciación y promoción, y puesta en valor del proyecto. Se puede encontrar más información en <http://www.ow2.org/> y <http://europe.ow2.org>.

Referencias

[1] OW2 Consortium. Europe Local Chapter, <http://europe.ow2.org/>.

[2] AspireRFID Project. <http://fp7-aspire.eu>.

[3] Stan Garfield. *Implementing a Successful KM Programme*. Ark Group Australia, 2007. <http://stangarfield.googlepages.com>.

[4] E. Wenger. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge University Press, 1998. ISBN: 0-521-66363-6.

[5] PLUME Project. *Promoting economicaL, Useful and Maintained softwarE for the Further Education And THE Research communities*. <http://www.projet-plume.org/en>.

[6] P. Goodyear, M. De Laat, V. Lally. Using Pattern Languages to Mediate Theory-Praxis Conversations in Designs for Networked Learning. *ALT-J, Research in Learning Technology. 1741-1629, Volume 14, Issue 3, 2006*, pp. 211 – 223.

[7] E. Wenger, R. McDermott, W. M. Snyder. *Cultivating Communities of Practice*. Harvard Business School Press, 2002. ISBN:1578513308.

[8] Karl Fogel. *Producing Open Source Software*, <http://producingoss.com/>.

[9] C. Latemann, S. Stieglitz. *Framework for Governance in Open Source Communities*, Postdam University, 2005. ISBN 0-7695-2268-8.

[10] OW2 Europe Local Chapter. Webinar Platform, <http://www.ow2.org/view/Activities/EuropeLocalChapterWebinars>.

[11] M. Gouvêa, C.Motta, F.Santoro. Recommendation as a Mechanism to Induce Participation in Communities of Practice, *UFRJ, UNIRIO, Brazil, 2006*. ISBN 978-3-540-32969-5.

[12] C. Ruggles, G. Wadley, M. Gibbs. Online Community Building Techniques Used by Video Game Developers. *Lecture Notes in Computer Science, 2005*. ISBN 978-3-540-29034-6. <http://www.springerlink.com/content/47dc4v3xej51p5m4/>.

[13] C.Cruz, M.Gouvêa, C.Motta, F. Santoro. A Proposal for Recognizing Reputation within Communities of Practice. *UFRJ, UNIRIO, Brazil, 2008*. ISBN 978-3-540-92718-1.

[14] P. Muller. Reputation, trust and the dynamics of leadership in communities of practice. *Journal of Management and Governance, 2006* ISSN 1385-3457 (Print) 1572-963X (Online).

[15] Fabernovel Consulting. *Modèles économiques des logiciels open source et logiciels libres* (2007) <http://www.fabernovel.com/businessmodels_opensource.pdf>.

[16] L. M. Sama, V. Shoaf. Ethical Leadership for the Professions: Fostering a Moral Community. *Journal of Business Ethics, 2007* ISSN 0167-4544 (Print) 1573-0697 (Online).

[17] McDermott Consulting. Communities of Practice, <http://www.mcdermottconsulting.com/communitypractice.shtml>.

[18] Audiofanzine. French music webzine, <http://fr.audiofanzine.com/>.

[19] JRES 2009. *Journées réseaux*, <https://2009.jres.org/>.

[20] Linux Online. WorldWide Linux User Groups <http://www.linux.org/groups/>.

[21] PICOM. *Pôle des industries du commerce*, Lille-France, <http://www.picom.fr/>.

Dinamización de comunidades en proyectos de software libre

Martin Michlmayr
Open Source Program Office, Hewlett-Packard

<tbm@hp.com>

Traducción: Teófilo Romera Otero (GSyC/LibreSoft, Universidad Rey Juan Carlos)

Resumen: *la creciente complejidad de las comunidades establecidas alrededor de proyectos de software libre junto con un aumento del interés de la explotación comercial de los mismos, ha llevado a que cada vez más de estas comunidades estén incorporando un puesto de "dinamizador de la comunidad". El dinamizador de la comunidad actúa como enlace entre empresa y comunidad y se asegura de que haya una buena relación entre los dos. También promueve la comunidad, ayuda a su crecimiento y cuida de que permanezca sana. Este artículo discute la dinamización de comunidades en proyectos de software libre y ofrece una visión sobre el papel del dinamizador de comunidades.*

Palabras clave: *colaboración, comunidad, dinamización, software libre, voluntarios.*

Autor

Martin Michlmayr ha estado participando en varios proyectos de software libre desde hace más de 10 años. Ha sido coordinador voluntario del proyecto GNUstep, Director de Publicidad de Linux International y líder del proyecto Debian. Durante los dos años al frente de Debian, Martin representaba al proyecto y llevaba a cabo importantes tareas de coordinación y organización. Martin ingresó en Hewlett-Packard en 2007 donde trabaja como experto en comunidades de software libre y dinamizador de la comunidad de FOSSBazaar. Martin también forma parte del Consejo de Open Source Initiative (OSI). Es diplomado en filosofía, sicología e ingeniería del software y doctor por la universidad de Cambridge.

1. Introducción

Las comunidades en Internet están reclamando cada vez más atención por diversas razones. Una de ellas es la enorme abundancia de conocimiento que los usuarios que participan en estas comunidades han creado de forma colaborativa. Internet ha jugado un importante papel en la creación de este conocimiento al posibilitar que gente con fines comunes se encuentre y compartan sus pasiones e intereses. Según Lessig [1], las oportunidades de colaboración que ofrece Internet han llevado a una transformación de una cultura predominantemente de sólo lectura a una de lectura y escritura. En otras palabras, menos individuos permanecen como consumidores pasivos y cada vez más gente comparte sus ideas de manera activa y crean nuevo conocimiento o productos. En lugar de meramente ver televisión, los aficionados usan Internet para discutir acerca de un espectáculo con sus amigos; además de leer libros, los fans escriben sus propias versiones de ficción o remezclas musicales y las comparten con otros.

Incluso en Internet, la mayoría de usuarios son consumidores pasivos pero hay una tendencia hacia una dedicación más activa. Esto ocurre en parte porque los sitios web se diseñan cada vez más con el objeto de atraer aportación y participación activa de usuarios y clientes. O'Reilly [2] utiliza el término *arquitectura de la participación* para describir "*la naturaleza de los sistemas diseñados para la participación de los usuarios*".

Hay muchos ejemplos de sitios en Internet que han utilizado esta arquitectura de la participación para atraer una base significativa de colaboradores activos. Wikipedia es la afamada enciclopedia que "*cualquiera puede editar*", Flickr es un sitio para compartir fotos que almacena incontables fotos de alta calidad (muchas de ellas bajo licencias Creative Commons), Amazon dispone de una abundante cantidad de reseñas de libros contribuidas por los propios lectores y quién no ha oído hablar de YouTube, que permite "*emitirse a uno mismo*". Lo que hace especiales a estos sitios, no es la tecnología subyacente que puede ser replicada fácilmente por cualquiera, sino la comunidad de usuarios que han atraído, que contribuye al sitio y hacen del sitio lo que es.

Reconociendo la importancia de las comunidades en Internet, Seth Godin, un conocido experto en marketing, identificó el papel del organizador de comunidades en Internet (o dinamizador de comunidades[1]) como uno de los más importantes trabajos del futuro [3]. Los dinamizadores de las comunidades ayudan tanto a la creación de comunidades como al sustento y crecimiento de comunidades existentes. Este artículo discutirá la dinamización de comunidades y el papel de dinamizador de comunidades en el contexto de los proyectos de software libre, que se apoyan en comunidades de desarrolladores y otros participantes para crear software. En la sección siguiente se discute el fenómeno de la dinamización de comunidades y los diferentes puntos de vista del puesto de dinamizador de comunidades.

2. Una definición de dinamización de comunidades

Los proyectos de software libre tienen una larga historia de desarrollo y participación de comunidades. El paradigma del software libre fue iniciado por individuos que compartían el objetivo común de crear software que diera a los usuarios más libertades que el software propietario [4]. Aunque el software libre puede ser desarrollado por una compañía sin implicación por parte de participantes externos, la mayoría del mismo es, o bien desarrollado por una comunidad de voluntarios (que pueden o no estar recibiendo un salario por parte de una empresa con interés en el software), o por una empresa que lidera el desarrollo de manera colaborativa junto con una comunidad de usuarios interesados en mejorar ese software. La comunidad es una característica definitoria de la mayoría de proyectos de software libre, junto con la licencia y las metodologías de desarrollo. La literatura académica también asocia los métodos colaborativos en los que se basa el software libre con varios aspectos positivos, como los altos niveles potenciales de revisión [5] y la innovación a través de los usuarios[2] [6].

Aunque la comunidad siempre ha tenido un papel importante en los proyectos de software libre, la dinamización de comunidades a cargo de personas con dedicación permanente y el puesto de dinamizador de la comunidad en sí, es un fenómeno relativamente reciente. Esto es en parte debido a que el éxito del software libre ha dado lugar a un enorme crecimiento de su base de desarrolladores y usuarios y esta evolución de la comunidad puede asociarse a ciertas dificultades en su crecimiento que necesitan ser gestionadas [7]. Además, los modelos económicos "híbridos" que combinan economías comerciales y de compartición están siendo empleadas de forma creciente en el contexto del desarrollo del software libre, y las interrelaciones entre ambas han de ser gestionadas activamente. Lessig [1] define un híbrido como "*o bien una entidad comercial que busca impulsar el valor de una economía participativa, o bien una economía*

participativa que construye una entidad comercial para ayudar a sostener sus objetivos". Mozilla, un proyecto de software libre que desarrolla el popular navegador de Internet Firefox, es un ejemplo de esto último puesto que la organización que lo sostiene ha obtenido fondos sustanciales a través de diferentes asociaciones con diferentes entidades. Existen innumerables ejemplos de empresas cuyos modelos de negocio se basan en Linux y otros proyectos de software libre y que tienen intereses clave en asegurarse de que las comunidades relacionadas se mantengan sanas.

Hay numerosas razones por las que las empresas quisieran establecer grandes comunidades alrededor de los proyectos de software libre que construyen. Además de obtener participantes que ayudan en el desarrollo y hacen evolucionar el software, el tener una comunidad le da a la gente un lugar en el que crear un vínculo con la empresa [8]. Esto, a su vez, puede llevar a una cierta lealtad a la marca, innovación de productos, y usuarios que pueden llegar a ser defensores de la compañía y sus productos y que pueden crear marketing viral. Red Hat (con Fedora) y Novell (con OpenSUSE) son dos buenos ejemplos de empresas de la industria alrededor de Linux que han establecido proyectos dirigidos por comunidades de los que las empresas se benefician de diversas maneras.

Los beneficios de las comunidades y la necesidad de promoverlas y gestionarlas han propiciado la figura del dinamizador de comunidad. Mientras que tradicionalmente el líder de proyecto se encargaba de que existiera una comunidad fuerte alrededor de su proyecto, actualmente las responsabilidades a menudo se reparten entre diferentes personas. El líder de proyecto es típicamente visto como una figura de tipo técnico que tiene la responsabilidad de hacer progresar el software y de ocuparse de los desarrolladores. El dinamizador de la comunidad, por otro lado, se asegura de que hay una comunidad sana en el proyecto, interactúa con los usuarios, los desarrolladores y otros actores y coordina aspectos organizativos del proyecto. Cuando una empresa es la fuerza motriz que impulsa un proyecto, el dinamizador de la comunidad actúa como enlace entre la compañía y la comunidad y se asegura de que hay una buena relación entre las dos.

Muchos proyectos de software libre exitosos que son dirigidos por empresas o que tienen altos niveles de implicación empresarial, tienen en estos momentos un dinamizador de la comunidad dedicado. Mientras muchos de ellos utilizan el título de "dinamizador de la comunidad"[3] , en realidad hay diferencias sustanciales entre las diferentes visiones existentes de este papel. Adam Williamson [9], antiguo empleado de Mandriva y actualmente trabajador de Red Hat, hizo la siguiente observación: *Es muy interesante observar las diferencias existentes en cómo el rol es percibido; Yo era una especie de portavoz de la comunidad de usuarios en Mandriva, Jono [Jono Bacon, dinamizador de la comunidad de Ubuntu] considera "comunidad" esencialmente a la comunidad de desarrolladores/colaboradores y Zonker [Joe Brockmeier, dinamizador de la comunidad de OpenSUSE] dice que su papel acaba siendo más que nada evangelización.*

Esta observación muestra que las responsabilidades del dinamizador de la comunidad son muy amplias. Tienen que tratar con los usuarios, los desarrolladores, los colaboradores y otros actores. A menudo actúan como defensores y promotores y llevan a cabo muchas otras tareas (las más importantes serán discutidas en el apartado siguiente). Una de las funciones clave de un dinamizador de la comunidad es asegurarse de que todo el mundo puede contribuir de manera óptima. Greg DeKoenigsberg [10], arquitecto de la comunidad de Red Hat, recalcó en una entrevista que "*es una figura que se centra en hacer la vida de los colaboradores individuales tan productiva y tan sencilla como sea posible*". DeKoenigsberg comenta sobre el papel de dinamizador de la comunidad: *Quieres hacer crecer la base de colaboradores, como sea. Quieres que tus desarrolladores principales se orienten a la comunidad, por supuesto. Pero también quieres que pasen la mayor parte de su tiempo haciendo lo que mejor saben hacer, que es (o eso espero) programar. Esto significa que quieres que sea alguien más quien se preocupe de mantener tu comunidad de colaboradores feliz y creciente.*

Asegurarse de que los colaboradores están a gusto y sacan partido de su participación en el proyecto es importante porque de otra manera podrían dejar de contribuir y dirigir sus energías a otros proyectos. En cualquier caso, esto lleva a preguntarse qué factores y valores son importantes para los voluntarios en proyectos colaborativos. Mark Shuttleworth, fundador del proyecto Ubuntu, sugiere que los voluntarios cuentan con encontrar las siguientes condiciones [1]:

- *Respeto: la comunidad y sus colaboradores han de ser respetados y considerados como corresponde.*
- *Responsabilidad: los colaboradores de la comunidad deben tener suficiente autoridad en el proyecto. Esto es particularmente importante en proyectos liderados por compañías puesto que tienen que equilibrar ese liderazgo con adjudicar a los colaboradores externos autoridad e influenciar en la dirección del proyecto.*
- *Significado: los colaboradores quieren tener la sensación de que el proyecto para el que contribuyen tiene sentido, "quieren sentir que forman parte de algo que es grande, importante y bonito". Si un proyecto no da a los voluntarios la sensación de que sus aportaciones y el proyecto en sí mismo tienen significado, invertirán su tiempo en un proyecto que sí lo haga.*

Estos valores son importantes porque están fuertemente relacionados con la motivación de los voluntarios y la única manera de influenciar el trabajo voluntario es a través de motivación. Jim Grisanzio [11], antiguo dinamizador de la comunidad de OpenSolaris, hizo algunos comentarios en su blog acerca de la distinción entre poder y liderazgo que son importantes en este contexto: *No se puede sustituir liderazgo con poder. Los líderes de voluntarios obtienen cooperación apelando a los valores compartidos de la comunidad y eso es un ejercicio mucho más exigente que dictar ordenes con amenazas o forzar a la gente a echarse atrás.*

En otras palabras, los dinamizadores de la comunidad no tienen autoridad sobre las actividades de colaboradores voluntarios (no pagados) y tienen que encontrar mejores modos que el poder para animarlos a trabajar en ciertas actividades. Han de liderar la comunidad escuchando sus necesidades y ayudando al mismo tiempo.

En resumen, la figura de dinamizador de comunidades es un desafío que incluye muchas tareas y responsabilidades diferentes. Los dinamizadores de la comunidad interactúan con varios actores y llevan a cabo muy diversas actividades. No tienen poder sobre los voluntarios y han de encontrar incentivos interesantes, como ofrecer un significado al proyecto, si quieren motivar a los voluntarios.

3. Tareas y aptitudes para la dinamización de comunidades

Como puede deducirse de la descripción de la figura de dinamizador de la comunidad expuesta anteriormente, un dinamizador de la comunidad ha de llevar a cabo varias y diversas actividades, incluyendo:

- Habilitación: una de las más importantes responsabilidades del dinamizador de comunidades es la de habilitar a la comunidad de diferentes maneras. Por ejemplo, el dinamizador de la comunidad ha de ayudar a que esta crezca mediante la captación de nuevos miembros para el proyecto y ayudándoles a iniciarse. Además, el dinamizador de la comunidad mantendrá en mente el "panorama general" del proyecto de manera que pueda poner en contacto a la gente. Si alguien ha trabajado o está trabajando en la solución a un problema que alguien más está sufriendo, pero no se conocen el uno al otro, el dinamizador de la comunidad podría ponerlos en contacto para que colaboren. Saber qué está pasando en todo el proyecto también es importante porque permite al dinamizador de

la comunidad sugerir actividades desatendidas a nuevos miembros del proyecto. La comunicación en general es muy importante tanto en pequeños como en grandes proyectos y es tarea principal del dinamizador de la comunidad el hacer circular noticias y mantener a la gente informada de lo que ocurre en el proyecto. Finalmente, resulta crucial asegurar interacciones saludables dentro de la comunidad. Muchos proyectos, especialmente cuando llegan a cierto tamaño, atraerán *trolls* que aportan energía negativa y hacen perder tiempo a los colaboradores. Algunos proyectos han adoptado un Código de Conducta que gobierna las comunicaciones y las interacciones dentro del proyecto. Es responsabilidad de todo el proyecto que este Código de Conducta sea respetado pero el dinamizador de la comunidad debería intervenir si nadie más lo hace.

- Delegación: como se ha tratado anteriormente, la habilitación resulta una responsabilidad clave. No es responsabilidad del dinamizador de la comunidad el llevar a cabo todas las tareas del proyecto. De hecho, el dinamizador de la comunidad debería tener la precaución de no ser demasiado activo en algunas áreas para no desanimar la participación de voluntarios potenciales. Por ejemplo, si un proyecto tiene un foro de soporte, el dinamizador de la comunidad no debería responder inmediatamente a todas las preguntas. En su lugar, es deseable que anime a otros voluntarios a ayudar y que traslade las preguntas a desarrolladores expertos que podrían contestarlas fácilmente. El dinamizador de la comunidad debería asegurarse de que el foro sirve de ayuda y puede por supuesto contestar preguntas de vez en cuando, especialmente si nadie más lo hace.
- Marketing: los dinamizadores de la comunidad a menudo llegan a jugar un papel de promotores. Esto podría implicar viajar a conferencias para hablar sobre un proyecto y animar a nuevos voluntarios a que se unan. También puede incluir hablar con la prensa sobre ciertos aspectos del proyecto. Finalmente, los dinamizadores de la comunidad son a menudo encargados de mantener buenas relaciones con otros proyectos y empresas.
- Escucha y observación: una de las ventajas clave de tener una comunidad grande es recabar sugerencias y opiniones de un gran número de personas con puntos de vista diferentes. Es responsabilidad del dinamizador de comunidades escuchar las necesidades y las ideas de los colaboradores y asegurarse de que esas reflexiones llegan a la gente adecuada dentro del proyecto o dentro de la empresa que lo promueve. Además de escuchar a la comunidad, el dinamizador de comunidades ha de comprobar que las sugerencias son tenidas en cuenta y que se hace algo al respecto de ellas.
- Creación de una visión: como ya se ha expuesto, los voluntarios necesitan una motivación clara para contribuir a un proyecto en lugar de hacerlo a otro. El dinamizador de la comunidad ha de ayudar a crear una cultura común para la comunidad así como una visión del proyecto. Es importante recoger sugerencias de los colaboradores y tenerlas en cuenta a la hora de hacer avanzar la comunidad en una dirección determinada.
- Información: el dinamizador de la comunidad ha de estar al tanto e informar a la empresa sobre el estado de salud de la comunidad. Al mismo tiempo ha de utilizar esta información con miras a mejorar la comunidad.

Basándonos en estas actividades, las aptitudes que se pueden derivar como vitales para un dinamizador de comunidades son varias:

- Comunicación: el dinamizador de la comunidad es un comunicador. Ha de interactuar con usuarios, desarrolladores, colaboradores y otros actores interesados o implicados en el proyecto. Ha de poner al día a la gente de lo que está ocurriendo en el proyecto y trabajar junto con representantes de otros proyectos, compañías o con la prensa. Es esencial tener habilidades para la escritura y muy recomendable saber hablar en público. Mas aun, puesto que muchos proyectos grandes son normalmente internacionales, es muy útil que los dinamizadores de comunidades conozcan y sepan como tratar con otras culturas de modo que colaboradores de otras culturas se sientan bienvenidos al proyecto.
- Marketing: dar a conocer el proyecto es muy importante para atraer nuevos colaboradores y potencialmente también para atraer otros recursos, como financiación. Aunque no es habitual que el dinamizador de la comunidad haga marketing en el sentido tradicional, no es raro que se le haga responsable del "marketing de la comunidad".
- Coordinación: organizar actividades varias es otra de las responsabilidades clave para los dinamizadores de comunidades y por eso han de ser buenos coordinadores. Por ejemplo, puede que tengan que organizar eventos tales como reuniones de usuarios, realizar el seguimiento de tareas que han de ser llevadas a cabo e identificar quién está al cargo de cuáles.
- Habilidades técnicas: aunque el dinamizador de comunidades no es una figura técnica, siempre ayuda saber algunos detalles técnicos. Esto es especialmente útil para entender problemas técnicos o retos dentro del proyecto y así poder ayudar a nuevos voluntarios a encontrar el área en que pueden participar mejor. Las habilidades técnicas pueden también ayudar cuando el dinamizador de la comunidad quiere montar un foro nuevo o cualquier otra infraestructura para el uso de la comunidad y el resto de la gente está demasiado ocupado con otras cosas.
- Paciencia y buena disposición: el dinamizador de la comunidad ha de escuchar a muchos colaboradores e interactuar con gente que es nueva en el proyecto y por tanto no está familiarizada con sus prácticas y su cultura. Además, hacer crecer la comunidad puede llevar mucho tiempo [12]. Es por esto que la paciencia y la buena disposición son virtudes importantes para un dinamizador de la comunidad.
- Estar visible y estar presente: como ya se ha comentado, el dinamizador de la comunidad no debería contestar todas las preguntas del foro o resolver los problemas de todo el mundo. Pero ha de estar disponible y visible de manera que los colaboradores sepan que siempre tendrán alguien a quién recurrir si tienen un problema. Foster [13] argumenta que ser un dinamizador de comunidades no es "un trabajo de 9 a 5", en parte porque ha de responder inmediatamente cuando hay una crisis y en parte porque las comunidades internacionales a menudo engloban muchas zonas horarias.

4. Actividades para crear comunidad

Existen muchas maneras de construir y hacer crecer una comunidad. Citamos a continuación varias actividades comunes empleadas por proyectos de software libre para dar una visión general del tipo de actividades que resultan adecuadas.

- Reuniones cara a cara: mientras que el trabajo diario en proyectos de software libre se lleva a cabo típicamente a través de Internet, las reuniones cara a cara han mostrado ser muy útiles por varias razones. Una de ellas es que son muy efectivas, especialmente cuando dan lugar a una lluvia de ideas o a resolver problemas conocidos. Otra razón es que las reuniones presenciales permiten que gente, que normalmente interactúa por Internet, se conozca en persona. Esto puede ayudar mucho a la colaboración futura, motiva a la gente y ayuda a crear un sentimiento de comunidad más fuerte.
- Stands en conferencias: pueden servir para conseguir nuevos voluntarios y permiten la interacción entre desarrolladores y usuarios. Ir a conferencias y ferias puede ser una buena motivación para los desarrolladores porque pueden conocer a la gente que usa su software, y también para muchos usuarios resulta grato conocer a los desarrolladores del software que utilizan. Encontrarse con los usuarios es también muy buena manera de obtener sugerencias para mejorar el software.
- Formación en Internet: varios proyectos han empezado recientemente a organizar eventos especiales para recién llegados que estén interesados en conocer más acerca del proyecto y de cómo contribuir. Por ejemplo, la *OpenSUSE Community Week* consiste en varias sesiones de formación acerca de diversos aspectos de OpenSUSE. La formación se imparte a través de IRC (Internet Relay Chat), un sistema de chat a menudo utilizado por

proyectos de software libre como medio vital de comunicación.

- Reuniones de caza de errores[4]: otra manera de facilitar que los recién llegados participen y de conseguir que algunos voluntarios colaboren de manera regular es organizar eventos (presenciales o a través de Internet) durante los cuales se analizan y reparan errores en el software. Estos eventos son a menudo coordinados por algún experto del proyecto en el control de calidad[5] y junto a muchos otros desarrolladores experimentados que ayudan a los nuevos colaboradores a empezar arreglando su primer fallo en el código. Esto puede convertirse en una muy buena motivación para los novatos que bien podrían decidir involucrarse más en el proyecto.
- Boletín de noticias: publicar anuncios y noticias sobre el proyecto de manera regular es una estupenda manera de mantener a la comunidad informada sobre el desarrollo del proyecto. El boletín de noticias puede además ser utilizado para publicar peticiones de ayuda o para promover actividades que los usuarios pudieran encontrar interesante.

Como se ha mencionado anteriormente, estos son sólo algunos ejemplos de actividades que los proyectos de software libre utilizan con éxito para construir sus comunidades. De algún modo, las actividades que saldrán bien dependerán de las particularidades del proyecto. Pero parece sensato observar otros proyectos para obtener ideas de actividades que pueden llevarse a cabo.

5. Conclusiones

Los proyectos de software libre se han apoyado siempre en comunidades de colaboradores que mejoran el software. Debido al incremento del tamaño de las comunidades y del interés comercial en las mismas, el papel de dinamizador de la comunidad está ganando relevancia. Un dinamizador de la comunidad puede actuar como el enlace entre la empresa y la comunidad y puede asegurarse de que haya una buena relación entre las dos. La figura del dinamizador de la comunidad comprende actividades diversas, desde la habilitación hasta la promoción. Este amplio conjunto de responsabilidades se refleja en las aptitudes que un dinamizador de la comunidad debería poder poner sobre la mesa, incluyendo muy buenas habilidades para la comunicación y la organización así como paciencia y buena disposición.

Referencias

[1] Lawrence Lessig. *Remix*. Bloomsbury Academic, London, 2008. ISBN: 978-1408113479.
[2] Tim O'Reilly. *The Architecture of Participation* (2004). <http://www.oreillynet.com/pub/a/oreilly/tim/articles/architecture_of_participation.html>.
[3] Seth Godin. Jobs of the future, #1: Online Community Organizer (2007). <http://sethgodin.typepad.com/seths_blog/2007/07/jobs-of-the-fut.html>.
[4] Richard Stallman. The GNU Operating System and the Free Software Movement. In: DiBona, C., Ockman, S., Stone, M. (Eds.), *Open Sources: Voices from the Open Source Revolution*. pp. 53-70 (1999).
[5] Eric S. Raymond. *The Cathedral and the Bazaar*. O'Reilly & Associates, 1999. ISBN-10: 1565927249.
[6] Karim R. Lakhani, Eric von Hippel. How open source software works: 'free' user-to-user assistance. *Research Policy, 32(6)*, 923-943 (2003).
[7] Martin Michlmayr, Benjamin Mako Hill. Quality and the Reliance on Individuals in Free Software Projects. *Proceedings of the 3rd Workshop on Open Source Software Engineering*, 105-109 (2003).
[8] Dawn Foster. *Why Companies Should Have Online Communities* (2008). <http://fastwonderblog.com/2008/04/16/why-companies-should-have-online-communities/>.
[9] Adam Williamson. *LinuxFest recap* (2009). <http://www.happyassassin.net/2009/04/27/linuxfest-recap/>.
[10] Greg DeKoenigsberg. Interview with Greg DeKoenigsberg, Red Hat Community Architect. *LinuxQuestions.org* <http://www.linuxquestions.org/questions/interviews-28/interview-with-greg-dekoenigsberg-red-hat-community-architect-725426/>.
[11] Jim Grisanzio. *The Distinction Between Power and Leadership* (2009). <http://blogs.sun.com/jimgris/entry/the_distinction_between_power_and>.
[12] Matthias Stürmer. *Open Source Community Building*. Master thesis, University of Bern, 2005.
[13] Dawn Foster. *What Does it Take to Manage a Community?* (2007). <http://fastwonderblog.com/2007/09/03/what-does-it-take-to-manage-a-community/>.

Notas del traductor

[1] En el original se utiliza el término *community manager*, que hemos optado por traducir como *dinamizador de la comunidad* en la versión en castellano.
[2] Del original *user innovation*, se refiere al término acuñado por Eric von Hipple para describir la innovación llevada a cabo por los usuarios y consumidores finales en lugar de por los proveedores.
[3] Nuevamente,, *community manager* en el original.
[4] En el original *Bug squashing parties*. Hace alusión al doble sentido de la palabra en inglés *bug* que generalmente se traduce como bicho pero además sirve como sinónimo de error en argot informático. Por otro lado, el uso que se hace de *squash* en este contexto, significa aplastar. De modo que la frase original se podría haber traducido literalmente como "comitiva de aplastamiento de bichos". Pero lo que se describe es una reunión en la que los desarrolladores dedican a buscar y solucionar errores en el código.
[5] De las siglas QA (*Quality Assurance*) en el original.

La Comunidad Morfeo: estrategias Open Source para la Open Innovation

Cristina Breña, Andrés Leonardo Martínez Ortiz
Telefónica Investigación y Desarrollo

<{crisbb,almo}@tid.es>

Resumen: *Morfeo es una comunidad de Software Libre, cuyos miembros son grandes empresas, administraciones públicas, universidades y centros de investigación, centros tecnológicos y pequeñas y medianas empresas. Todos ellos comparten el principio de que el Software Libre es una estrategia efectiva para mejorar los procesos de investigación, desarrollo e innovación de la tecnología (software). El software de plataformas permite definir modelos de negocio viables y relaciones win-win que impulsan la colaboración. MyMobileWeb, ezWeb y las contribuciones de CENATIC y OpenFwPA, framework de e-Administración del Principado de Asturias, son casos de éxito de producción tecnológica en la Comunidad Morfeo. No obstante, la mejora de la comunicación interna/externa, el desarrollo de nuevas herramientas que soporten la colaboración entre empresas y la búsqueda de soluciones para otros problemas complejos de colaboración empresarial tales como la producción de estándares o el desarrollo empresarial, constituyen retos actuales de la Comunidad Morfeo.*

Palabras clave: *fomento empresarial, forjas, innovación abierta, software libre.*

Autores

Cristina Breña es estudiante de último curso de periodismo en la Universidad Rey Juan Carlos. Durante su periodo de estudiante, ha trabajado como locutora de radio y ha obtenido becas para el estudio en el extranjero (Helsinki) en 2007. Actualmente, está haciendo prácticas de periodismo en Telefónica I+D, ayudando en tareas de comunicación. Está a cargo de elaborar estudios de viabilidad a través de redes sociales, escribir notas de prensa y artículos, y presentar mejoras y refuerzos del plan de Comunicación de la Comunidad Morfeo.

Andrés Leonardo Martinez Ortiz es Licenciado en Informática por la Universidad Politécnica de Madrid. Desde 2007 es especialista en OSS (*Open Source Software*) en Telefónica I+D. Durante el periodo entre 1998 y 2002 fue ingeniero del departamento de I+D de Teldat, un fabricante de infraestructuras y desde 2002 hasta 2007 fue miembro activo del grupo de investigación GSyC/LibreSoft de la universidad Rey Juan Carlos, donde era profesor. Empezó en software libre cuando era ingeniero de I+D en Teldat, donde portó el *kernel* de Linux a la arquitectura M860. En la actualidad, es miembro del Proyecto Morfeo, donde se encarga de la dinamización de la comunidad y participa en iniciativas como OSS marketplace o la definición de programas para la certificación y la formación de tecnologías OSS. Participa en Vulcano y Qualipso que tienen objetivos similares: promover la adopción del software libre en la industria. Como resultado de esta participación, está definiendo y montando el Centro de Competencia de Morfeo, que se incluye en la red de centros de competencia de Qualipso, y gestiona el desarrollo de nuevas herramientas de colaboración. Forma parte de grupos de trabajo en software libre de INES OSS y NESSI OSS, la asociación IEEE y ACM.

1. Introducción

Una de las mayores contribuciones a la revolución tecnológica del siglo XX ha sido la popularidad que han alcanzado los métodos de colaboración colectiva. Internet y algunos de sus servicios más famosos, como Flickr, Wikipedia o Facebook, son resultados de la cooperación. Asimismo, estos servicios se han convertido en paradigmas colaborativos [1][2]. Mucho antes, comunidades de código abierto como Debian, Linux kernel, Apache, Mozilla u OpenOffice ya habían definido el modelo de colaboración para el desarrollo del software y produjeron la infraestructura requerida [3][4] para soportar una comunidad de desarrolladores de ámbito mundial.

No obstante, existen todavía muchos problemas complejos [5] cuya solución pasa por la definición de mecanismos colaborativos entre los agentes interesados en resolverlos. Por ejemplo, la gestión de la investigación científica, el desarrollo tecnológico y, aún más complicado, los procesos de innovación continúan sin ser resueltos. Así, podemos encontrar todavía algunos escenarios donde parece difícil definir modelos de colaboración, como por ejemplo las colaboraciones entre empresas, donde la competitividad parece inherente al mercado.

Es en este contexto, en la búsqueda de soluciones para promover la productividad, la ventaja competitiva y las estrategias efectivas para competir en el mercado del software, donde surgió la Comunidad de software libre Morfeo en la que están implicados todos los interesados en los procesos de I+D+i.

El proyecto Morfeo [6] es un ecosistema tecnológico y económico compuesto por empresas, administraciones públicas, universidades y grupos de investigación, centros tecnológicos y grupos industriales, y pequeñas y medianas empresas (PYMES). Todas estas organizaciones están implicadas en la investigación, el desarrollo y la innovación de la producción tecnológica, y comparten los siguientes objetivos:

- La promoción de estándares abiertos y plataformas de software.
- La definición de modelos de negocio basados en las tecnologías licenciadas como abiertas.
- La promoción de los mercados competitivos de provisión tecnológica como medio para lograr soluciones innovadoras, producción tecnológica más barata y evitar situaciones de oligopolio/monopolio.

Cada miembro tiene su propia motivación (beneficio derivado de la cooperación) para participar en las actividades de la comunidad, como por ejemplo:

- Las universidades y grupos de investigación pueden complementar (más o menos) sus investigaciones con actividades de transferencia tecnológica realizadas por grandes empresas y PYMES.
- Las administraciones públicas, como grandes consumidoras de tecnología y a veces también proveedoras de ella, necesitan una gran base de proveedores/clientes y mejorar sus posiciones para definir estándares de administración electrónica.
- Las empresas consiguen masas críticas en sus inversiones tecnológicas reduciendo el riesgo de adopción tecnológica.
- Pequeñas y medianas empresas pueden participar en la última parte de la cadena de valor disminuyendo el riesgo de crear nuevos servicios y productos. En otras palabras, las PYMES pueden acceder a las actividades de I+D+i y vencer sus restricciones financieras.

2. El proyecto Morfeo

A finales del año 2004, el Proyecto Morfeo fue creado conjuntamente por Telefónica Investigación y Desarrollo (TID), una compañía

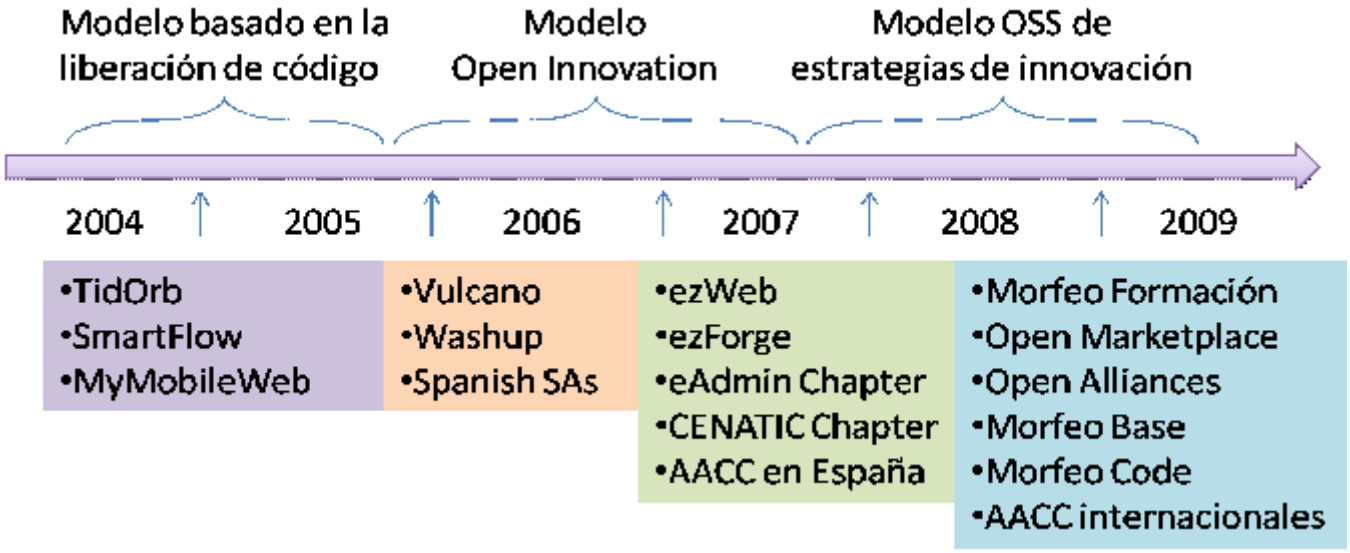

Figura 1. Evolución del modelo de la Comunidad Morfeo.

que pertenece al grupo Telefónica, la Universidad Politécnica de Madrid (UPM) y la Universidad Rey Juan Carlos (URJC). La iniciativa persiguió demostrar la viabilidad no sólo en el uso del software de código abierto en las industrias sino también en la participación de las compañías en el desarrollo del software de código abierto. La evolución de la comunidad muestra cómo fueron tomados diferentes enfoques para alcanzar los objetivos de Morfeo (ver **figura 1**).

Desde mitad del año 2004 hasta el final del 2005: liberación de código

Durante ese tiempo, Telefónica I+D fue ayudado por la URJC y la UPM en el proceso de liberación de proyectos de software internos (privativos) para convertirlos en proyectos de código abierto. Fueron liberados; TidOrbJ, una implementación en Java de un Object Request Broker de Corba, SmartFlow, un motor de estados basado en redes de Petri, y MyMobileWeb, un marco para desarrollar contenidos web para dispositivos móviles. Todos ellos eran productos maduros con una hoja de ruta cercana al final e intensamente validados dentro del grupo Telefónica. Los principales retos fueron construir alrededor de esos productos una comunidad (tradicional) dirigida principalmente a desarrolladores voluntarios. Para hacer esto, por ejemplo, se creó una forja de comunidad [7], que es una herramienta colaborativa para el desarrollo del software.

Hoy en día, se puede decir que el éxito de esta estrategia no ha sido uniforme. Mientras que para TidOrb y SmartFlow fue imposible crear una comunidad, MyMobileWeb ha tenido un éxito destacable ya que en torno a esta tecnología existe una comunidad pequeña, pero activa formada por desarrolladores en todo el mundo. La conclusión principal que obtenemos de esta etapa es que es muy difícil mezclar en la misma comunidad socios industriales, sin experiencia en el desarrollo de software de código abierto, con miembros de la comunidad OSS (*Open Source Software*) tradicional, con importantes prejuicios hacia las compañías y sus necesidades e intereses.

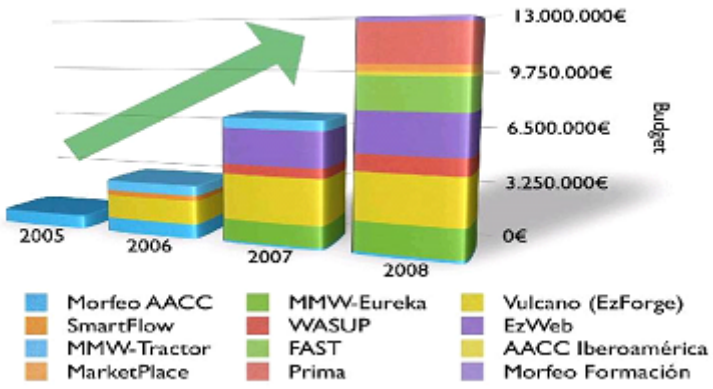

Figura 2. Evolución del presupuesto total de Morfeo (© 2008 Javier Soriano).

Desde 2006 hasta el final del 2007: construir una comunidad de innovación abierta

En la mitad del 2005, el *Board* de la Comunidad Morfeo se dio cuenta de las dificultades de mezclar la visión tradicional del OSS con la industria en Morfeo. El principal problema fue un desconfianza mutua: los usuarios tradicionales de software de código abierto no tenían confianza acerca de la intención de la industria y cómo esta podría contribuir a la comunidad de OSS; por otra parte, la industria tuvo problemas para adoptar la tecnología de código abierto tanto en el soporte de sus procesos de negocio como en la entrega de servicios y productos, principalmente debido al poco conocimiento sobre la tecnología OSS y su proceso de desarrollo. Por lo tanto, se optó por cambiar de estrategia: construir una comunidad de innovación abierta [8] para organizaciones que participan en procesos de I+D+i. Así, Morfeo empezó a construir un ecosistema de investigación, desarrollo e innovación tecnológica con suficiente calidad como para ser aceptada bajo criterios industriales. La respuesta fue inmediata: los implicados en I+D+i encontraron un esquema para resolver el riesgo en la actividades relacionadas con la administración tecnológica. Como resultado, Morfeo tuvo que definir nuevas reglas para fomentar la participación con nuevos canales de comunicación y nuevas infraestructuras. También, por ejemplo, se hizo una remodelación de la comunidad adaptada a las nuevas clases de miembros que empezaron a formar parte de ella. Esta vez los resultados fueron bastante distintos. Desde 2006 hasta ahora, Morfeo ha incubado muchas propuestas de I+D+i. Muchas de ellas han tenido financiación pública y el total del presupuesto ha estado en torno a los 25 millones de euros. El progreso de los presupuestos se puede ver en la **figura 2**.

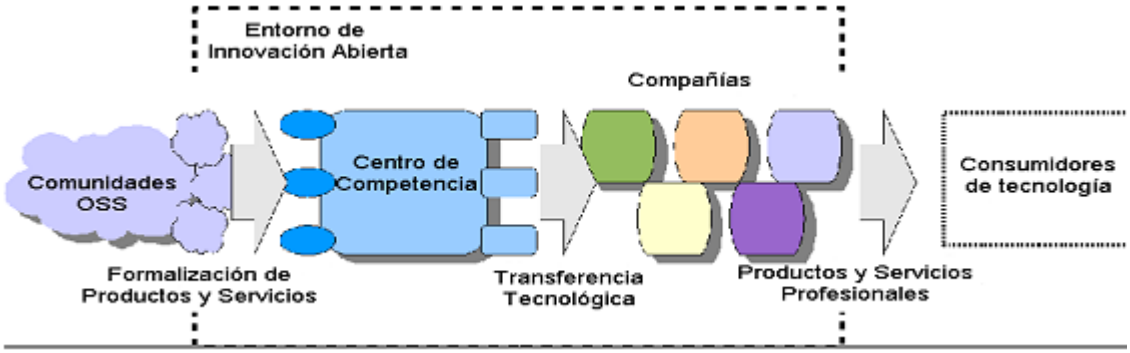

Figura 3. Centro de Competencia de Morfeo.

Desde 2008 hasta ahora: Estrategia OSS para la innovación

Desde luego, la historia previa ha mostrado cómo la estrategia de software de código abierto puede ser usada para crear un esquema de colaboración no competitiva entre las organizaciones. Por lo tanto, fue lógico abordar nuevos desafíos para intentar aplicar estas estrategias a otros problemas complejos en las relaciones entre empresas. A finales de 2008, los miembros de Morfeo empezaron a trabajar en nuevos problemas: los procesos de estandarización, el desarrollo de servicios profesionales basados en la tecnología OSS y el desarrollo de facilidades en las relaciones empresariales para las PYMES. Estas tres iniciativas han recibido los siguientes nombres: Morfeo Open Alliance, Centro de Competencia Morfeo y Morfeo Base.

Morfeo Open Alliance es una iniciativa para crear un consorcio de empresas que compartan una visión común de las tecnologías y de sus arquitecturas asociadas. Además, este consorcio acuerda unas especificaciones abiertas y potencia las implementaciones de código abierto de los componentes en la arquitectura acordada. Este modelo de colaboración acelera la definición de estándares lo que se traduce en un periodo de tiempo más corto en la comercialización y en un bajo riesgo de inversión tecnológica. La iniciativa ha sido probada en un dominio específico: Service Front-end [9].

El Centro de Competencia Morfeo [10] es una iniciativa para fomentar el desarrollo de servicios profesionales basados en tecnologías OSS. El concepto del centro de competencia en OSS fue introducido por el proyecto QualiPSo [11] como un modo de "dirigir ayudas a las industrias y gobiernos en busca de abastecimiento innovador y competitivo, estimulando el uso de software de código abierto flexible y de bajo precio para desarrollar sistemas de información viables e innovadores". El centro de competencia Morfeo es una propuesta para ser incluida en la Red QualiPSo de centros de competencia. El centro de competencia pretende impulsar la adopción del OSS para fomentar el desarrollo de servicios profesionales de tecnologías de OSS que pueden ayudar a reducir el riesgo de la adopción de OSS en la industria. La **figura 3** muestra su estructura y modelo de negocio.

Finalmente, Morfeo Base [12] es la iniciativa para aumentar el desarrollo económico de las PYMES. Morfeo en esta propuesta está trabajando en la definición de las estrategias de modulación de las cadenas de valor bajo aproximaciones colaborativas. Como caso especial, la comunidad está tratando de desarrollar una estrategia específica basada en la *comoditización* de soluciones software (la Ley de Christensen [13]) y trasladar valores añadidos desde los productos (vendiendo licencias) hasta los servicios (pago por uso). Esto permitiría crear un ecosistema de las PYMES alrededor de las comunidades OSS.

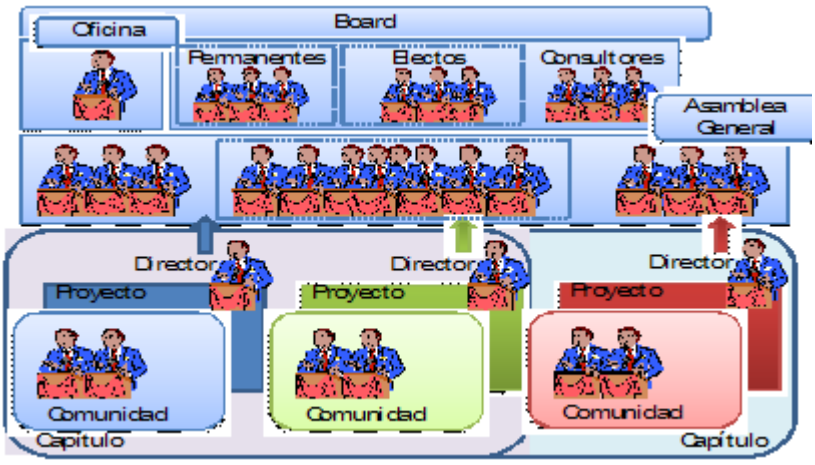

Figura 4. Modelo organizativo de Morfeo.

3. La estructura de una comunidad de innovación abierta

La estrategia actual del Proyecto Morfeo está basada en una estructura específica de comunidad. Esto es así porque la Comunidad Morfeo no es como el resto de las comunidades de código abierto. Morfeo no está compuesta principalmente por voluntarios y desarrolladores individuales de software sino por agentes de los procesos de I+D+i. También, la clase de instituciones que forman parte de la comunidad hacen de Morfeo la única comunidad con dicha estructura.

La comunidad está organizada en proyectos y capítulos (ver **figura 4**). Un capítulo no es más que una agrupación de proyectos que tienen alguna clase de relación entre ellos. Cada proyecto tiene sus propias reglas debido a que Morfeo es muy flexible en ese criterio. En el Proyecto Morfeo no hay que pagar ninguna cuota por ser miembro, tan sólo seguir estas tres condiciones:

1) Tecnología libre: los resultados del proyecto deben ser libres bajo el criterio OSI [14].
2) Gestión de la Propiedad Intelectual definida: Así, los contribuidores potenciales pueden conocer las condiciones de colaboración.
3) Fuerte mentor: La propuesta de proyecto deber ser liderada al menos por una organización como una empresa, universidad o grupo de investigación, una administración pública, un centro tecnológico o una PYME.

Incluso para aquellos proyectos que no cumplen con la tercera condición, Morfeo les ofrece una comunidad más tradicional conocida como Morfeo Bazaar. En cualquier caso, para estar en Morfeo, el "futuro" miembro debe enviar una solicitud de alta de proyecto en la que tiene que rellenar los objetivos y una

descripción clara del proyecto, el consorcio, la licencia y la política de gestión de propiedad intelectual. Después de las validaciones, el proyecto se convertirá en un proyecto Morfeo y los participantes en el consorcio del proyecto entrarán a formar parte de la Comunidad Morfeo.

Cada proyecto tiene un representante en la Asamblea General donde el *Board* informa cada año acerca del estado de la comunidad, comparten *roadmap* (hoja de ruta), nuevas propuestas de servicios horizontales, nuevos proyectos y miembros. El *Morfeo Board* es el comité ejecutivo de la comunidad formado por miembros permanentes, como por ejemplo, los miembros fundadores o permanentes, los representantes de proyectos elegidos para ser miembros del *Board* y consultores, que son profesionales prominentes de OSS con gran habilidad asesora. El *Board* está al cargo de identificar nuevas estrategias de OSS y de introducir nuevos servicios para fomentar la colaboración entre los miembros de Morfeo. Finalmente, la Oficina Morfeo es la encargada de las actividades diarias como son el mantenimiento de la infraestructura, la administración de los proyectos que se aprueban, las tareas de comunicación, etc. La oficina Morfeo apoya al *Board* dándole información para la toma de decisiones.

4. Produciendo tecnología de innovación abierta

Actualmente Morfeo tiene diferentes proyectos tecnológicos, reflejando la evolución de la comunidad. Así, podemos encontrar los siguientes tipos de proyectos:

- Proyectos heredados: MyMobileWeb [15] "es una plataforma de software de bajo-coste, modular, basado en estándar abiertos que simplifica el desarrollo de aplicaciones web de alta calidad para móviles y portales. Proporciona una mejora en los contenidos y en las adaptaciones de las aplicaciones". Inicialmente fue desarrollado por Telefónica I+D y liberado como un proyecto OSS cuando comenzó la comunidad. Después de eso, MyMobileWeb ha sido capaz de crear una comunidad alrededor suyo. Está liderada por Telefónica I+D y es una de los pocos casos de éxito de esta clase de proyectos.
- Proyectos de innovación abierta: EzWeb [16] es una plataforma *mash-up* que cuenta con mecanismos enriquecidos de comunicación entre *gadgets* y que permite construir el *front-end* de una nueva generación de arquitectura SOA. Es asimismo la propuesta de Morfeo de *front-end* de la nueva Internet del futuro. La plataforma ha sido desarrollada desde cero en el ámbito de la comunidad. Dentro del proyecto participan muchas organizaciones, principalmente PYMES, pero también trabajan con universidades y administraciones públicas. Este proyecto está liderado por Telefónica I+D aunque actualmente hay otras empresas que se están implicando en la tecnología EzWeb. Claramente se puede decir que éste es otro de los logros en los proyectos de la comunidad de innovación abierta.
- Proyectos comunitarios: el capitulo CENATIC [17] y OpenFWPA [18] son ejemplos de proyectos comunitarios. En estos casos las administraciones públicas o las agencias públicas establecieron un marco de trabajo con la Comunidad Morfeo. Ellos ofrecen claras estrategias de adopciones de OSS o tecnología OSS para la implementación de la administración electrónica, mientras que Morfeo ofrece una activa comunidad de organizaciones implicadas en procesos de I+D+i. Esta clase de proyectos no están liderados por ninguno de los miembros fundadores, sino por organizaciones externas que selecciona la Comunidad Morfeo para incubar esta clase de proyectos. Los proyectos comunitarios son un ejemplo del efecto expansivo que tiene la comunidad que se refleja en un entorno de soporte tecnológico.

5. Conclusiones

El Proyecto Morfeo ha conseguido algunos éxitos importantes en estos años. Morfeo ha sido capaz de definir estrategias efectivas de OSS para la gestión de la innovación. Se ha construido un entorno de innovación abierta donde la gestión de la propiedad intelectual abierta y basada en licenciamiento libre es un aspecto fundamental. Además, Morfeo es un referente en el programa público de I+D+i ya que ha llegado a ser un verdadero entorno de transferencia tecnológico, fomento empresarial y crecimiento económico. A pesar de que Morfeo comenzó siendo una iniciativa de ámbito español, actualmente, tiene un reconocimiento también europeo. La comunidad ha empezado a colaborar en Latinoamérica con la incorporación de una nueva red de oficinas en Iberoamérica (Brasil, Argentina, Uruguay, Paraguay y Chile).

Reconocimientos

Este artículo no hubiera sido posible hacerlo sin la experiencia, opiniones y comentarios de algunos de los principales contribuidores del Proyecto Morfeo. Ha sido un placer entrevistar a las siguientes personas: Pedro Acebes, Juan Antonio Cáceres, Jesús M. González, Álvaro Polo, Marcos Reyes y Roberto Santos.

Referencias

[1] D. Tapscott, A.D. Williams. *Wikinomics: How mass collaboration changes everything*. Portfolio Hardcover, 2006. ISBN-10: 1591841380.

[2] C. Shirky. *Here comes everybody: The power of organizing without organizations*. The Penguin Press HC, 2008. ISBN-10: 1594201536.

[3] E.S. Raymond. *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly, 2001. ISBN-10: 0596001312.

[4] K. Fogel. *Producing Open Source Software: How to Run a Successful Free Software Project*. O'Reilly Media, 2005. ISBN-10: 0596007590.

[5] Software Engineering Institute, Carnegie Mellon University. *Ultra-Large-Scale Systems: The Software Challenge of the Future*. <http://www.sei.cmu.edu/uls/files/ULS_Book2006.pdf?bcsi_scan_87A666907766D0F0=0&bcsi_scan_ filename=ULS_Book2006.pdf>.

[6] Morfeo Project. <http://www.morfeo-project.org>.

[7] Morfeo Forge. <http://forge.morfeo-project.org>.

[8] H. W. Chesbrough. *Open Innovation: The New Imperative for Creating and Profiting from Technology*. Harvard Business School Press, 2005.

[9] Morfeo Open Coalition for Service Front Ends. <http://sfe.morfeo-project.org>.

[10] Morfeo Competence Center. <http://cc.morfeo-project.org>.

[11] QualiPSo Project. <http://www.qualipso.org>.

[12] Morfeo Base. <http://base.morfeo-project.org>.

[13] C.M. Christensen, M.E. Raynor. *The Innovator's Solution: Creating and Sustaining Successful Growth*. Harvard Business School Press, 2003. ISBN-10: 1578518520.

[14] Open Source Initiative. Section about Open Source Definition and Open Source Licenses <http://www.opensource.org>.

[15] MyMobileWeb Project. <http://mymobileweb.morfeo-project.org>.

[16] ezWeb Project. <http://ezweb.morfeo-project.org>.

[17] Morfeo Capítulo CENATIC. <http://cenatic.morfeo-project.org/>.

[18] OpenFwPA Project. <http://openfwpa.morfeo-project.org>.

Aplicación de los principios del software libre en líneas de producto

Frank van der Linden
Philips Healthcare, Países Bajos

<frank.van.der.linden@philips.com>

Traducción: Tomás Aguado Gómez (GSyC/LibreSoft, Universidad Rey Juan Carlos)

Resumen: *la Ingeniería de Líneas de Producto mejora la gestión de la variabilidad y la reutilización de sistemas empotrados. Ayuda a explotar la personalización en masa con el objeto de presentar al cliente un producto individualizado. Sin embargo la Ingeniería de Líneas de Producto también ha llevado a procesos de desarrollo muy densos y a la necesidad de una planificación global para un rango completo de productos (una línea de producto). El desarrollo de software libre no ha tenido un impacto profundo en las compañías de desarrollo de sistemas empotrados, en parte debido a que la cultura inherente al desarrollo de software libre no se mantiene demasiado fiel al establecimiento de procesos de desarrollo pesados y de planificación global. Sin embargo ambos esquemas de desarrollo necesitan (a menudo) de desarrollo distribuido, y es en este punto donde ambos pueden aprovechar los logros del otro. Este artículo investiga las diversas opciones en las que el software libre y las metodologías de desarrollo asociadas pueden utilizarse para reducir los problemas que aparecen en el desarrollo distribuido así como para aumentar la calidad del software resultante. Una pequeña parte del artículo estudia las opciones disponibles para que el software libre se beneficie de las metodologías de gestión de la variabilidad de las líneas de producto software y de este modo aumentar las posibilidades de personalizar los productos resultantes.*

Palabras clave: *comercialización, fuente interna, gestión del cambio, líneas de producto software, software libre.*

Autor

Frank van der Linden trabaja en las oficinas de Philips Healthcare. Recibió su doctorado en Matemáticas en 1984 en la Universidad de Amsterdam y desde entonces ha trabajado para Philips. Sus intereses más recientes incluyen las metodologías de desarrollo software y arquitecturas, principalmente metodologías orientadas a componentes y objetos. En 1991 se implicó en el desarrollo de una Línea de Producto dentro de Philips, aspecto en el que ha trabajado hasta el día de hoy. Fue jefe de proyecto en tres proyectos ITEA sucesivos que versaban sobre Ingeniería de Líneas de Producto, y posteriormente en desarrollo distribuido incluyendo metodologías de desarrollo de software libre. Como parte de estos proyectos fue miembro del comité organizativo de talleres y conferencias sobre líneas de producto (PFE & SPLC). Entre 2005 y 2008 fue líder del proyecto ITEA COSI sobre desarrollo distribuido y prácticas de software libre en la industria de sistemas empotrados. En este proyecto ha organizado gran cantidad de talleres sobre Software Libre y Líneas de Producto. Es coautor de diversos libros y numerosos artículos sobre estos temas.

1. Introducción

En las últimas décadas, los proveedores de sistemas empotrados han introducido la Ingeniería de Líneas de Producto con el objeto de mejorar la gestión de la diversidad de sus productos así como para reducir el esfuerzo de desarrollo. El movimiento hacia la Ingeniería de Líneas de Producto normalmente tiene por objetivo la reutilización a gran escala, llevando a la reducción de costes y del período de comercialización así como a la mejora de la calidad y la reducción de los costes de mantenimiento.

La Ingeniería de Líneas de Producto Software es un medio de desarrollar aplicaciones software usando plataformas y personalizaciones masivas. Las personalizaciones en masa permiten la producción eficiente y rápida de un producto individualizado para el cliente. Los artefactos utilizados para los diferentes productos han de ser lo suficientemente adaptables para encajar en los diferentes sistemas que se producen en la línea de producto. Esto supone la necesidad de gestionar la variabilidad de todos los artefactos de la línea de producción. Específicamente una línea de producto generalmente define la plataforma de una arquitectura, sus componentes y las herramientas de apoyo para proporcionar una gestión del cambio (variabilidad) eficiente y estandarizada.

Otras tendencias en la Ingeniería del Software no están siempre relacionadas con líneas de producto software. Sin embargo, la aplicación de estas metodologías en una organización que las desarrolle podría ser beneficioso, Asimismo, otros enfoques de la Ingeniería del Software podrían beneficiarse de la introducción de principios basados en el desarrollo de líneas de Producto. Esto último aplica al desarrollo de software libre como un modelo válido y eficiente de desarrollo de software.

Existen diferentes opciones a la hora de desarrollar software libre. Sin embargo, el denominador común de todas es que un grupo distribuido de personas trabaja conjuntamente parar generar software. La principal diferencia con el enfoque basado en líneas de Producto es que las comunidades de desarrollo de software libre tienen motivaciones propias y específicas tratándose a menudo de desarrollos con el único objetivo de ofrecer solución a un problema determinado y de compartir esta solución con otros.

No existe un único beneficio a la hora de aplicar conceptos del software libre a la Ingeniería de Líneas de Producto. Del mismo modo, debido al diverso uso que se hace del software libre, la Ingeniería de Líneas de Producto puede ser atractiva para las comunidades de desarrollo de software libre. Sin embargo, muchas de las prácticas seguidas desde el enfoque de Ingeniería de Líneas de Producto no encajan con las seguidas en las comunidades de software libre. Por ejemplo los procesos *gestionados* no son siempre viables o aceptados. Siempre será beneficioso combinar las ventajas de ambos enfoques aunque esto no será posible si no se eliminan estas incompatibilidades. Actualmente, existe una interacción limitada entre las comunidades de software libre y las de desarrollo de líneas de producto, aunque este interés ya ha sido plasmado en anteriores estudios [3].
Este artículo investiga las relaciones entre el software libre y el desarrollo de líneas de producto. Se basa en los resultados obtenidos en el proyecto COSI de ITEA [5]. A partir de este estudio se pueden establecer las siguientes conclusiones:

- Implantar una línea de producto software implica realizar una gran inversión, planificaciones a muy largo plazo, gestión explícita del cambio y desarrollo *distribuido.*
- La industría informática usa el software libre como medio para obtener software de calidad de manera rentable. Esto ayuda a reducir costes compartiendo esfuerzos, y disminuye el tiempo de desarrollo del proyecto a través de metodologías ágiles de desarrollo; todos estos procedimientos constituyen en sí mismos una manera *distribuida* de concebir y desarrollar sistemas.
- El problema de la paralelización en el ámbito del desarrollo distribuido es la principal inspiración para intentar combinar las prácticas de Software Libre y de la Ingeniería de Líneas de Producto.

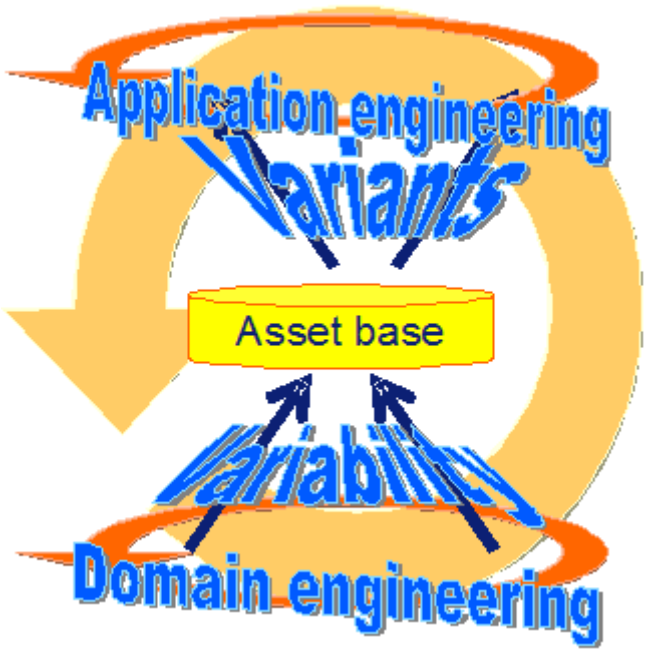

Figura 1. Procesos de Ingeniería de Líneas de Producto.

En este artículo se analizan los diferentes aspectos de la Ingeniería de Líneas de Producto, y se relacionan con prácticas de Software Libre. El artículo finaliza con un resumen de la aplicación de prácticas de Ingeniería de Productos en el desarrollo de Software Libre.

2. Conceptos básicos de Ingeniería de Líneas de Producto

2.1. Dos procesos de desarrollo

La Ingeniería de Líneas de Producto hace énfasis en la separación entre los conceptos *construir una plataforma robusta* y *crear en un corto espacio de tiempo aplicaciones personalizadas para el cliente.* Estos dos conceptos conducen a dos procesos interrealacionados de desarrollo. (ver **figura 1**):

- Establecimiento de la plataforma (Ingeniería de Dominio), incluyendo la definición de características comunes y de variabilidad de todos los elementos que componen la línea de producto.
- Derivación de aplicaciones (Ingeniería de Aplicación), incluyendo la vinculación de la variabilidad en las aplicaciones.

La separación en estos dos procesos también indica la separación de los dos problemas más importantes relacionados con la gestión del cambio. La Ingeniería del Dominio es responsable de asegurar que la variabilidad es la apropiada para el desarrollo de aplicaciones. La Ingeniería de Aplicación se focaliza en el desarrollo de sistemas individuales sobre la plataforma. Reusa la plataforma y vincula a las aplicaciones. Estos dos procesos están pensados para funcionar de manera desacoplada pero sincronizados por las implementaciones de la plataforma. Debido a que estos dos procesos están débilmente acoplados pueden ser gestionados por modelos de ciclo de vida totalmente diferentes.

2.2. Variabilidad

La Ingeniería de Líneas de Producto Software [16] [11] tiene por objetivo apoyar un amplio espectro de productos, orientados a diferentes clientes específicos. En vez de comprender cada sistema individual por sí mismo la Ingeniería de Líneas de Producto Software observa la línea de producto como un todo y sobre éste la variación de todos los sistemas individuales. Igualmente importante es conocer las características comunes de todos los componentes que se gestionan en una línea de producto. A través de esta Ingeniería se deben gestionar tanto las características comunes como la variabilidad. La variabilidad se define mediante la Ingeniería de Dominio y se explota durante la Ingeniería de Aplicación vinculando las variantes apropiadas. La variabilidad se introduce a lo largo de muchas fases durante el proceso de Ingeniería de Dominio.

En cada nivel se refina la variabilidad del nivel anterior y se introducen características adicionales que no son resultado de este refinamiento. Este proceso se describe en la **figura 2**. La variabilidad se define inicialmente durante el análisis de las necesidades de todas las entidades implicadas para ser posteriormente refinada a lo largo de las siguientes fases dle proceso. A esto se le denomina *variabilidad interna*. Las necesidades específicas de cada fase llevan a la introducción de nuevas variabilidades internas.

Establecer la infraestructura de una línea de producto no es un objetivo en sí mismo. El objetivo primordial es su explotación a lo largo del proceso de Ingeniería de Aplicación. A esto también se le denomina *instanciación* de la variabilidad. Los siguientes términos, que describen entidades autocontenidas en todo tipo de artefactos de desarrollo, son importantes en la gestión de la variabilidad:

- *Punto de Variación:* Describe dónde se encuentran las diferencias entre los sistemas finales.
- *Variante:* Las diferentes posibilidades que existen para satisfacer un *punto de variación.*

En la mayoría de los casos los puntos de variación no cambian independientemente. La selección de una variante específica para un punto de variación dado, influye en las opciones posibles para otros puntos de variación. Para permitir la gestión de la variabilidad, es necesario un modelo de variabilidad para asegurar que se realiza una elección consistente de variantes a lo largo de todos los puntos de variación. En propuestas tempranas de modelado de variabilidad, este modelado se integraba con la notación subyacente (utilizando por ejemplo la herencia). Sin embargo, se admite comunmente que es necesario establecer una distinción clara entre

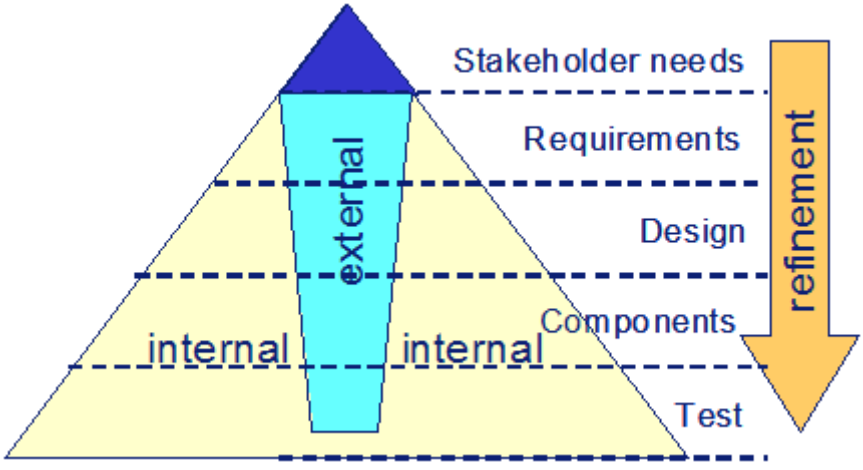

Figura 2. Pirámide de Variabilidad.

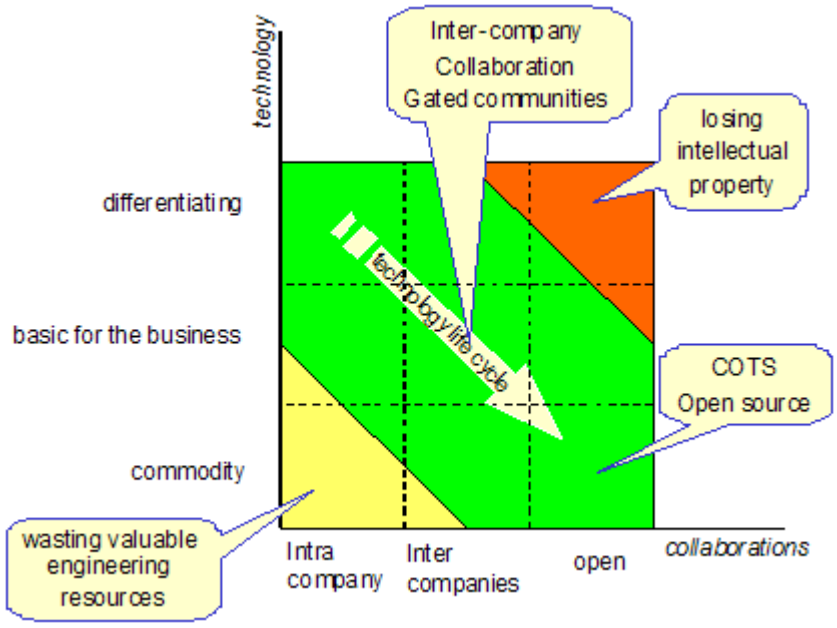

Figura 3. Desarrollo eficiente y efectivo de software.

el modelo de variabilidad y el del propio sistema. Esto hace mucho más sencillo aplicar ajustes complejos, da un apoyo mejorado a la toma de decisiones y es mucho más escalable [1].

Existen diversas propuestas de modelos de variabilidad. Los grupos más importantes son los Modelos de Variabilidad basados en Árboles de Características [10] y los Modelos de Variabilidad Ortogonal (OVM) [16] .

También debe distinguirse entre variabilidad en el tiempo y en el espacio [2]. Ambos tipos de variabilidad son posibles con puntos de variación y variantes diseñados y aplicados de manera apropiada. La variabilidad en el tiempo está asociada a la evolución de la línea de producto y representa la existencia de diferentes versiones de un artefacto que son válidas en diferentes lapsos de tiempo. Esto último está relacionado con el versionado de los elementos del sistema. La variabilidad en el espacio esta referida a la existencia de un artefacto en diferentes formas dentro del mismo espacio de tiempo. Esto lleva a la existencia de diferentes configuraciones que conforman sistemas válidos al mismo tiempo. Estas prácticas están relacionadas directamente con la Ingeniería de Líneas de Producto.

2.3. Software libre aplicado a líneas de producto

El interés de las empresas por el software libre tiene su origen en el reconocimiento del hecho de que la práctica totalidad del software se está convirtiendo en una materia prima de primera necesidad, lo que hace que sea muy interesante para los desarrolladores de sistemas empotrados introducirlo tanto en sus productos como en las líneas de producto asociadas.

3. La mercantilización del software

Como consecuencia de la *mercantilización* del software [13] gran parte del mismo ha dejado de estar particularizado para un producto concreto. De hecho actualmente no es común que un producto software esté desarrollado por una única compañía. Por el contrario, el software se produce en estrecha colaboración dentro de la propia empresa y más allá de sus fronteras. Más aun, es muy común que software de terceros se integre como parte del producto propio.

Para la mayoría de los productos y líneas de producto, únicamente una pequeña parte[1] (5%-10%) del software es diferenciador. Esta pequeña parte es la que proporciona valor añadido sobre el producto ofrecido por otras empresas competidoras. El resto del software es más o menos común en el dominio, o incluso entre diversos dominios. El desarrollo de software eficiente y efectivo únicamente se concentra en la producción de módulos *diferenciadores*. El software como mercancía (la parte común) puede y debería ser adquirido a terceros, implicando desarrollo distribuido y software de terceros como COTS (*Commercial Off The Shelf*) o software libre. Como la Ingeniería de Dominio está más influenciada por el software puede necesitar diversos caminos para ajustarse a una aplicación específica.

La **figura 3** muestra un esquema confrontando la tecnologías contra las decisiones empresariales de construir o adquirir software. Las dos esquinas marcadas en la figura deberían ser evitadas a la hora de desarrollar tecnología. La esquina superior derecha debería ser evitada a cualquier precio ya que implicaría regalar el valor añadido propio a las empresas competidoras. La esquina inferior izquierda también debería ser evitada para ahorrar recursos de desarrollo debido a que la parte común de la tecnología puede ser obtenida de manera mucho más barata comprándola en vez de desarrollándola.

Un desarrollo software saludable está caracterizado por hallarse en el área central de la figura. El software diferenciador se desarrolla dentro de la propia organización y el software que da funcionalidades comunes se adquiere en el mercado o se dispone de él sin coste (software libre).

Cualquier software, incluyendo el desarrollo de líneas de producto, se desplaza de arriba hacia abajo a lo largo de la **figura 3**. El software comienza siendo diferenciador para alguna de las partes implicadas. Más adelante, en un momento dado ese software no proporciona el suficiente valor añadido a los productos y pasa a la consideración de "básico para el negocio". En etapas posteriores el software llega a convertirse en *commodity* (mercadería sin valor diferencial).

El desarrollo coherente de software también se caracteriza por moverse de izquierda a derecha al mismo tiempo, desde desarrollo interno a colaboraciones abiertas entre empresas. Para evitar alcanzar las situaciones indeseadas que se han descrito es necesario realizar estos movimientos de apertura al ritmo adecuado. Un ejemplo de este movimiento es el caso del software DVTk de Phillips Healthcare[2] [13] que evolucionó en varias etapas desde software propietario y diferenciador a software libre y *commodity* entre 1994 a 2002.

Cada compañía necesita analizar el software que desarrolla con respecto a la **figura 3** con el objeto de conocer cuándo cambiar la manera en la que colabora con otras compañías, esto es de especial importancia para las compañías que desarrollan líneas de producto. Debido a la larga vida de una línea de producto, hay partes de la misma que constantemente se convierten en *commodity*, mientras que la propia línea de producto ha de mantenerse viva y próspera. Esto requiere un esfuerzo adicional durante la fase de planificación y definición de recursos. Una complicación adicional es el hecho de que las compañías tienen un control limitado sobre el software libre que utilizan en sus desarrollos.

4. Software libre para líneas de producto

Debido al tamaño del desarrollo, las organizaciones implicadas en desarrollos de líneas de producto suelen optar por el desarrollo distribuido que tan eficientemente ha funcionado en las comunidades de desarrollo de software libre. Además, a la hora de reutilizar software es inteligente no ignorar la gran

cantidad de software libre disponible. Dentro del proyecto COSI [5] experimentamos con casos de estudio en los que se aprovechaba el software libre de 5 maneras diferentes:
1) Adopción de las prácticas de desarrollo dentro del desarrollo de la línea de producto (desarrollo interno).
2) Uso de herramientas de software libre en el desarrollo de la línea de producto. Ésta es la forma más sencilla de usar software libre en líneas de producto.
3) Uso de componentes de software libre en los productos de la línea de producto. Esto implica una mayor participación y planificación.
4) Liberación de productos de la línea de productos. Productos o líneas completas de producto que fueron inicialmente desarrollados de manera interna se liberan en comunidades de software libre.
5) Establecimiento de una relación simbiótica. Desarrollando la línea de producto usando los recursos de una comunidad de software libre.

4.1. Fuente interna

La *fuente interna* es una manera de aprovechar las ventajas del desarrollo distribuido de la manera en que lo hace el software libre, pero con el deseo de evitar problemas en la planificación, la propiedad y el control del proyecto. Diversas compañías han adoptado modelos de desarrollo basados en *fuente interna* [9].

En el modelo de desarrollo de fuente interna un conjunto de equipos colaboran en un ecosistema cooperativo. De manera similar al desarrollo de software libre el desarrollo de fuente interna aplica un modelo de cooperación abierto y concurrente. Este modelo implica la propiedad y el control distribuido, publicaciones tempranas y frecuentes de las aplicaciones desarrolladas, y canales eficientes y abiertos para realimentación por parte de todos los grupos implicados.

Se hace uso de los mecanismos de organización que ya están implantados en la empresa, por ejemplo los designados para la gestión de conflictos, definición de tareas, etc. El modelo de fuente interna proporciona flexibilidad a la hora de gestionar las colaboraciones (establecerlas, cancelarlas y modificarlas) y también es muy útil a la hora de coordinar equipos y definir sus prioridades a través de fronteras organizacionales y geográficas. Las compañías pueden usar este modelo como un paso intermedio para la integración de software libre en sus productos.

El modelo de desarrollo de fuente interna es el establecido dentro de Phillips Healthcare [19]. La compañía entrega una línea de producto que implica un amplio conjunto de productos de imagen médica a sus clientes (generalmente hospitales) en varias modalidades. El grupo de Ingeniería de Dominio proporciona una plataforma compuesta de un conjunto de componentes reusables y reconfigurables basados en una arquitectura común. Este grupo abordó el problema de convertirse en un cuello de botella de los diferentes grupos de Ingeniería de Aplicación. Las entregas del grupo de Ingeniería de Dominio se planifican en un proceso con gran cantidad de personas implicadas, por lo que el calendario de entregas no puede satisfacer siempre la planificación de todos los grupos de Ingeniería de Aplicación. Las dinámicas del mercado pueden forzar a cambios en la planificación de ciertos grupos de productos y sin embargo el proceso de planificación del dominio carece de la flexibilidad para adaptarse a estos cambios.

El desarrollo de *fuente interna* está basado en elementos del desarrollo de software libre, apoyados por un entorno web de desarrollo abierto y colaborativo. Desacopla la Ingeniería de Aplicación de la Ingeniería de Dominio ya que cada grupo de Ingeniería de Aplicación puede decidir por sí mismo sobre la plataforma:

- *Usarla tal cual.* Esperar a la siguiente entrega de la plataforma.
- *Contribuir con parches.* Obtener una versión anterior de un componente y aplicar parches de manera opcional.
- *Trabajar juntos como un equipo virtual.* Asumir la responsabilidad del desarrollo para los componentes de dominio cruciales para el grupo.

El principio más importante del desarrollo de fuente interna es el de contar con un acceso sencillo a toda la información de la línea de producto. Del mismo modo que en el caso del software libre la política deberá ser publicar rápido y a menudo para favorecer que los flujos de información entre Ingeniería de Dominio y Aplicación sean lo más fluidos posible. Los equipos de Ingeniería de Dominio poseen y desarrollan los componentes de dominio. A un desarrollador de aplicaciones se le permite modificar componentes de dominio, pero es responsable de dicho cambio. El cambio que se realiza podrá ser ofrecido "de vuelta" a los equipos de Ingeniería de Dominio. El equipo encargado de la gestión de la plataforma podrá aceptar el parche como parte de la propia plataforma, haciéndose responsable a partir de ese momento de dicho cambio.

Como consecuencia de esto, la implicación de los equipos de Ingeniería de Aplicación en lo que respecta al dominio se mejora. Esto tiene como efecto colateral que la plataforma se usa de manera más extensa que antes y se ajusta mejor a las necesidades de los grupos de Ingeniería de Aplicación. La metodología de fuente interna permitió la publicación de nuevos productos desde 2005 y la reducción del tiempo de comercialización en al menos 3 meses.

Sin embargo este modelo no ayuda a introducir software de terceros en la línea de producto. La fase de colaboración se limita a la propia empresa. El control del software es en este caso muy claro y el desarrollo distribuido y el mantenimiento se mejora sobre la situación original. La metodología de fuente interna proporciona métodos de colaboración para el desarrollo de software sobre métodos tradicionales.

4.2. Usando herramientas de software libre

Esta es la forma más sencilla de introducir software libre en líneas de producto. Como el software libre no aparece en el producto final, es muy sencillo para el usuario cumplir con las reglas de la comunidad de software libre. Muchas herramientas está ya disponibles como software libre, aunque un factor en contra es que no existen muchas herramientas de software libre específicamente desarrolladas para la gestión de líneas de producto. Una lista de este tipo de herramientas puede encontrarse en las páginas de distribuidores de software para líneas de producto[3] . La mayoría de estas herramientas no son software libre. Algunas herramientas que se utilizan en el proyecto COSI para el desarrollo de líneas de producto y que son software libre son:

- Stylebase para Eclipse: Herramientas de software libre que sirven para compartir y reutilizar conocimiento sobre arquitecturas software.[4]
- Subversion: Proporciona gestión de versiones.[5]
- Semantic MediaWiki: Apoya el desarrollo colaborativo de documentos dentro de un entorno de desarrollo colaborativo.[6]

4.3. Usando componentes de software libre

El uso de componentes de software libre en líneas de producto no es un principio muy diferente al de usar cualquier otro software de terceros (COTS) en la línea de producto. Sin embargo existen diversos puntos sobre los que es necesario prestar especial atención. La planificación de la introducción de software de terceros en una línea de producto siempre se ve afectada por el hecho de que el ritmo de desarrollo de este software no está controlado por la propia compañía que lo utiliza: nuevas versiones de la aplicación que mejoran versiones anteriores de modos que no están controlados por la empresa que lo utiliza. Estas mejoras influyen en la línea de producto, ya que partes de este software de terceros se usan dentro de la Ingeniería de Dominio afectado a todos los productos. Por lo tanto esta nueva versión deberá ser incorporada y las partes *propietarias* del software deberán ser adaptadas. A menudo la nueva versión se basa en diferentes mecanismos de arquitectu-

ra e interfaces con respecto a los que están en uso en la línea de producto. Para solventar este problema la empresa que gestiona la línea de producto necesita saber con antelación cómo va a evolucionar el software de terceros. En el caso de COTS esto implica tener un contacto fluido con el proveedor. En el caso de un desarrollo de software libre la empresa deberá implicarse en la propia comunidad de desarrollo, o al menos estar conectada a las listas de correo en las que se discuten todos estos aspectos. Además, si alguno de los desarrolladores de la empresa se implica en la comunidad de desarrollo, la empresa podría influir en el modo en el que este software va a evolucionar, pudiendo incluso llevar a la incorporación de nuevas arquitecturas o estándares en el producto de software libre.
En lugar de esperar a nuevas versiones del COTS y a mantener contacto con el proveedor, el software libre se adapta de manera mucho más continua. La posibilidad de obtener versiones tempranas del nuevo software (aunque pudieran aun tener algunos errores) sería de gran ayuda para comprobar que los nuevos cambios son conformes a la arquitectura de la línea de producto, así como para acelerar la incorporación de la primera versión estable. De modo recíproco el envío de informes de fallos y correcciones ayudará a mejorar la calidad del software que se desarrolla por parte de la comunidad de desarrollo, que si bien no es ya un elemento diferenciador para la empresa, es de vital importancia para la línea de producto. Se ha de decidir cuidadosamente qué partes (y qué versiones) de software libre se convertirán en parte de la plataforma, y qué software libre será únicamente usado en ciertas aplicaciones. De hecho, podría convertirse en una estrategia incorporar las últimas versiones de software libre únicamente en algunas aplicaciones *de prueba*, como es el caso del uso de COTS.

Otro problema en el desarrollo de software libre está relacionado con las licencias. Este problema podría afectar al modo en el que el software libre puede ser usado dentro de una línea de producto. Algunas licencias de software libre harían necesario publicar las fuentes del código adicional (el que realmente da valor añadido). Hay opciones técnicas para solventar este problema prestando especial atención a mantener un acoplamiento débil entre el software libre que se integra y el desarrollado de manera interna en la empresa. Sin embargo probablemente acabará siendo necesario que la empresa done alguna parte de su software a la comunidad de desarrollo para demostrar su buena voluntad hacia ella [14].

Es necesario tomar precauciones cuando se incorpora software de distintas fuentes en la línea de producto ya que las licencias particulares de cada componente podrían entrar en conflicto. Este aspecto deberá gestionarse dentro de la propia organización por las siguientes razones:

- Ignorar las cuestiones relacionadas con las licencias en alguno de los departamentos podría llevar a la necesidad de liberar software de otros departamentos de alto valor competitivo para la empresa.
- El uso de software libre en el desarrollo de aplicación podría indicar un movimiento hacia la mercantilización, lo que a su vez originaría que ese software debería formar parte del software común. Hay que tener en cuenta que esto no es siempre lo que se pretende, ya que el software libre debería ser utilizado únicamente en código específico de aplicación para ciertos productos.
- Nuevas versiones de los componentes basados en software libre podrían provocar que algunos módulos de las aplicaciones quedaran obsoletos.
- El uso de software libre podrían llevar a estándares internos adaptados en la línea de producto. El único modo de gestionar este problema es a través del acuerdo de todas las partes implicadas en el desarrollo de la línea de producto.

En el contexto en el que la empresa se implica en el desarrollo de software libre, ésta podría introducir en la comunidad de desarrollo sus propias capacidades de gestión de la variabilidad. Esto facilitaría una solución para el problema de la gestión de la configuración en las comunidades de software libre. Sin embargo la propia comunidad podría no valorar esta aportación y descartar las capacidades de gestión de la variabilidad. En este caso, la empresa podría asociarse con otra comunidad que sí adopte estas herramientas. Esto podría asegurar el mantenimiento de los mecanismos que gestiona el aplicativo de software libre utilizado y garantizaría la aceptación por parte de la comunidad de software libre. No obstante es necesario tener en cuenta que el control sobre estos mecanismos podría perderse una vez son liberados. La empresa deberá mantener su implicación en la comunidad y por ello invertir en personal o económicamente en la comunidad de desarrollo con el objeto de mantener el estado operativo de las herramientas.

En caso de conflictos con estándares de arquitectura, una opción es usar el conocimiento adquirido para COTS. Crear coberturas propietarias o código de cohesión que conecte los módulos internos de código con los que se encuentran en el mundo del software libre podría implicar actualizaciones demasiado frecuentes debido a que ambos módulos evolucionan de manera independiente [14].

Otra opción está basada en que la empresa adopte la misma solución arquitectónica que se usa dentro de la comunidad de desarrollo de software libre. La principal desventaja es que tanto la Ingeniería de Aplicación como la de Dominio deberán asumir esta solución arquitectónica, lo que podría suponer un esfuerzo demasiado grande. La empresa ha de asegurar en este caso que los mecanismos de la comunidad de software libre se adaptan a sus necesidades internas, y que a través de ellos pueden gestionar la variabilidad inherente a la línea de producto.

4.4. Liberando los productos de la línea de producto

Muchas empresas han pasado por experiencias muy satisfactorias a la hora de liberar software mercantilizado. A pesar de que en primera instancia pueda parecer que se está regalando la propiedad intelectual de los módulos que se liberan, no suele ser el caso. El software que se encuentra en la parte inferior del diagrama de la **figura 3** no suele contener propiedad intelectual valiosa para la empresa y por esta razón puede ser liberado sin correr un riesgo excesivo. Un recurso más importante es el conocimiento sobre cómo adaptar el software propio a las necesidades de los clientes. Éste es un conocimiento que en ningún caso debería ser liberado. Los clientes son conscientes de que es mucho más sencillo y barato contratar a la empresa para instalar el software y adaptarlo que asumir por sí mismos la configuración completa.

Para Philips Healthcare la primera experiencia liberando software es la suite DVTk[7] [13]. Este software da soporte a la verificación de la conformidad de imágenes médicas con el estándar DICOM. Como los clientes conectarán equipamiento proporcionado por diferentes proveedores es importante que todas las compañías cumplan con el estándar. Sin embargo, DVTk es únicamente software de base sin valor comercial *per se* para la compañía. Por esta razón Philips decidió en 2005 liberarlo lo que permitió compartir los esfuerzos de desarrollo y mantenimiento a una escala mucho mayor. De manera específica llevó a un desarrollo y mantenimiento mucho más rápidos por parte de la compañía, especialmente en aquellas partes de DICOM que son genéricas y no específicas de las necesidades de Philips.

4.5. Relaciones simbióticas

Finalmente, los módulos propios del software de base pueden ser liberados a la comunidad para asegurar que ésta los mantiene y apoya su posterior evolución. Esto lleva a una reducción drástica de los costes de mantenimiento de la empresa, ya que éstos pasa a compartirse con competidores y personas de otros dominios que también necesitan el software. Las mejoras y pruebas masivas del software serán realizadas por la comunidad y la propia compañía apoyará el desarrollo para asegurar que los problemas que aparezcan se solventan.
Esto puede ser particularmente efectivo para herramientas o componentes que han sido

obtenidos por otras vías. La implicación en la comunidad permite a la empresa asegurarse de que todos los requisitos internos se consideran y resuelven, y que el software sigue la línea evolutiva más beneficiosa para la empresa. Por ejemplo, Philips Healthcare es ahora parte de la comunidad Subversion[8]. El objetivo inicial era solventar los problemas que aparecían en la resolución de conflictos en el árbol de versiones. Este enfoque fue de gran éxito a través de la donación de una herramienta específica que solventaba este problema y que se incluyó como parte de la versión 1.6 de Subversion. Mientras tanto Philips se ha convertido en usuario experto y miembro activo de la comunidad Subversion influyendo realmente en la evolución de este proyecto.

5. Líneas de producto en desarrollo de software libre

5.1. Prácticas de líneas de producto

Desde la perspectiva del software libre, algunas investigaciones han explorado proyectos de software libre con el objeto de analizar cómo ciertos principios y prácticas comunes de líneas de producto han sido adoptadas por éstos. Por ejemplo, un análisis del núcleo de Linux *"demuestra cómo el núcleo de Linux ha alcanzado algunos de los objetivos que las guías de las Líneas de Producto Software proponen"* [17]. Más allá, van Gurp [7] analizó las prácticas usadas en tres grandes proyectos de software libre (Eclipse, Mozilla y el *kernel* de Linux) desde una visión orientada a sugerir mejoras en el desarrollo de líneas de producto. Este estudio concluye con una sugerencia a los propietarios de líneas de producto software *"hay un conjunto de prácticas que pueden ser encontradas en gran cantidad de proyectos de software libre que han probado su funcionamiento satisfactorio al menos en este contexto"*.

Chastek et al. [4] han investigado los modelos de desarrollo de líneas de producto desde una visión orientada a analizar cómo han adoptado este modelo proyectos específicos de software libre. Han usado un *framework* para pruebas de software de líneas de producto[9] desarrollado por el Instituto de Ingeniería de la Universidad Carnegie Mellon® (SEI). En sus análisis del proyecto Eclipse y su comunidad, demostraron que *"Eclipse ha conseguido alcanzar un equilibrio estable entre la dirección planificada y las contribuciones individuales; también comparte el reto de la Ingeniería de Líneas de Producto de gestionar de manera efectiva la comunicación entre aquellos que desarrollan recursos clave y los que los utilizan"*. Resumiendo, a pesar de que algunos proyectos de software libre pueden ser analizados como una línea de producto (por ejemplo Eclipse), los principios y prácticas de línea de Producto no se usan de manera generalizada por parte de las comunidades de desarrollo de software libre.

5.2. Arquitectura y variabilidad

Un recurso muy importante en las líneas de producto es la arquitectura de la plataforma, que define estándares para toda la línea de producto. Todos los sistemas han de ser conformes a esta arquitectura para asegurar el uso efectivo de la plataforma y el alcance de los objetivos de las líneas de producto para reducir el esfuerzo de desarrollo. El problema principal es que la arquitectura a menudo no está correctamente definida en las comunidades de software libre y por esta razón la conformidad con la misma es difícil de comprobar [8].

Las comunidades de software libre tienen a menudo otros mecanismos para gestionar la variabilidad muy diferentes de los utilizados dentro de una línea de producto. En muchos casos el software libre nunca fue concebido para ser usado en líneas de producto, y los problemas relacionados con los procesos de configuración compleja son abundantes dentro del mundo del software libre. Los puntos de variación y las variantes están (casi) ausentes en las metodologías de desarrollo de software libre. La variación espacial y temporal no se diferencia de manera clara. La variación se modela tradicionalmente a través del código o de directivas de compilación, lo cual no resulta muy efectivo para las líneas de producto.

No obstante las comunidades de software libre han desarrollado métodos efectivos de tratar con la configuración interna y la construcción del producto final.

5.3. Dos procesos

Sin embargo dentro de las comunidades de software libre no existe una distinción clara entre la Ingeniería de Dominio y la Ingeniería de Aplicación, aunque ambas actividades se realizan dentro de las comunidades de software libre. Un grupo esencial de desarrolladores experimentados se ocupan de las actividades que podrían englobarse dentro de la Ingeniería de Dominio. El software principal que se usa en toda el aplicativo es desarrollado por estos desarrolladores principales. Los desarrolladores con menos experiencia o menos activos trabajan a menudo en aplicaciones específicas. Sin embargo, del mismo modo que en la Ingeniería de Líneas de Producto, la funcionalidad principal puede emerger también de estos desarrolladores. Promocionar un recurso desde una aplicación hacia el propio dominio es relativamente sencillo en el caso de desarrollo de software libre, y es también una de las ventajas del modelo de desarrollo de fuente interna.

El ámbito del software libre no está a menudo correctamente delimitado. Esto lo hace más flexible y amplía sus posibilidades de utilización. Sin embargo la falta de una arquitectura correctamente definida y el modelo de variabilidad utilizado hace difícil aplicar el software en un entorno específico. Esto lleva a conflictos a la hora de decidir qué mecanismos se permiten y cuáles no, reduciendo así la consistencia de los diferentes módulos.

6. Conclusiones

Este artículo ha analizado el uso de software libre en líneas de producto. El modelo de desarrollo de software libre es inherentemente un modelo distribuido que constituye un modo muy atractivo y probado de construir software de gran calidad. Las empresas pueden usar las comunidades de software libre para producir y mantener software de base (sin alto valor diferenciador). Esto libera recursos de las compañías para producir software que sí tenga un alto valor específico.

Sin embargo la introducción de software libre en este ámbito no está libre de problemas, y aunque se solventen es necesario mantener ciertos recursos involucrados en la comunidad de software libre. Como el control y la propiedad del software no quedan (en la mayoría de los casos) en la empresa que lo usa, ésta ha continuar invirtiendo recursos para aprovechar las ventajas del software libre. La empresa ha de considerar dentro de sus procesos de planificación cómo implicarse y como monitorizar la evolución del software libre que usa en sus líneas de producto, así como cuándo y cómo se introducen nuevas versiones. Deben tenerse también en cuenta los problemas legales que podrían aparecer si el grupo de Ingeniería de Dominio no tiene conocimiento de los recursos de la línea de producto que son software libre, así como de sus licencias. Problemas similares pueden aparecer si el equipo de Ingeniería de Aplicación acopla demasiado el código interno que aporta valor añadido a los módulos de software libre.

Podrían aparecer problemas si la arquitectura de los módulos de software libre es incompatible con alguno de los productos de la línea de producto. La empresa podría utilizar el poder adquirido a través de su implicación en la comunidad para proponer y llevar a cabo los cambios que adapten esta arquitectura a la suya o incorporar código que recubra estas características incompatibles.

Las empresas que usan líneas de producto cuentan a menudo con una gran organización distribuida. El desarrollo distribuido genera gran cantidad de problemas de coordinación, algunos de los cuales son solventados por el modelos de desarrollo de software libre. Con el objeto de solventar algunas de las desventajas del uso de software libre, puede utilizarse el modelo de desarrollo de fuente interna. Este último soluciona el problema del desarrollo distribuido, pero no ayuda a compartir el esfuerzo que se da cuando componentes de software libre se integran como parte de la línea de producto.

Referencias

[1] F. Bachmann, M. Goedicke, J. Leite, R. Nord, K. Pohl, B. Ramesh, A. Vilbig. "A Meta-Model for Representing Variability in Product Family Development", *Proceedings of the 5th International Workshop on Product Family Engineering (PFE-5), Siena, Italy*, 2003, pp. 66-80.

[2] J. Bosch, G. Florijn, D. Greefhorst, J. Kuusela, H. Obbink, K. Pohl. "Variability Issues in Software Product Lines". *Proceedings of the 4th International Workshop in Product Family Engineering (PFE-4), Bilbao, Spain, October 3-5, 2001*, Springer, Berlin Heidelberg New York, LNCS 2290, 2002, pp. 13-21.

[3] Jan Bosch. "The Challenges of Broadening the scope of Software Product Families". *Communications of the ACM, December 2006*, pp. 41-44.

[4] G.J. Chastek , J.D. McGregor, L.M. Northrop. "Observations from Viewing Eclipse as a Product Line". En *F. van der Linden, B. Lundell (Eds.) OSSPL07 Asia, September 10, 2007*, Kyoto, Japan <http://itea-cosi.org/modules/wikimod/index.php?page=OssPlas07>.

[5] COSI. Co-development using inner & Open source in Software Intensive systems, ITEA project 2005-2008, <http://itea-cosi.org/ and http://www.friprog.no/Laer-mer/Prosjekter/COSI-Library-of-Assets>.

[6] B. Fitzgerald. "The Transformation of open source Software". *MIS Quarterly, Vol. 30, No. 3*, 2006 pp. 587-598.

[7] J. van Gurp. "OSS Product Family Engineering". *First International Workshop on open source Software and Product Lines*, Maryland, 2006 <http://www.sei.cmu.edu/splc2006/Gurp_paper.pdf>.

[8] I. Hammouda, T. Mikkonen. Open source Contributions as Platform Specialization Units. En *F. van der Linden, B. Lundell, Proceedings on the Second International Workshop on OSSPL07 open source Software and Product Lines 2007, June 14, 2007*, Limerick, Ireland <http://www.itea-cosi.org/modules/wikimod/index.php?page=OssPl07 paper #3>.

[9] A.A. Jilderda. *Inner Source Software Engineering at MIP fostering a meritocracy of peers.* Research Report, Philips Research, The Netherlands, 2004.

[10] K. Kang, S. Cohen, J.A. Hess, W.E. Novak, S.A. Peterson. "*Feature-Oriented Domain Analysis (FODA) Feasiblity Study*", Technical Report, Software Engineering Institute, Carnegie-Mellon University, 1990.

[11] F. van der Linden, K. Schmid, E. Rommes. "*Software Product Lines in Action*", Springer Verlag, 2007

[12] F. van der Linden, B. Lundell. *Proceedings on the Third International Workshop on open source Software and Product Lines: OSSPL07 Asia, September 10, 2007,* Kyoto, Japan. <http://itea-cosi.org/modules/wikimod/index.php?page=OssPlas07>.

[13] F. van der Linden, B. Lundell, P. Marttiin."Commodification of Industrial Software: A Case for Open Source", pendiente de publicar *IEEE Software July-August 2009*.

[14] J. Merilinna, M. Matinlassi. "Product Family Approach for Integration of In-house Software and open source Components". En *F. van der Linden, B. Lundell, Proceedings on the Second International Workshop on OSSPL07 open source Software and Product Lines 2007, June 14 2007*, Limerick, Ireland <http://www.itea-cosi.org/modules/wikimod/index.php?page=OssPl07 paper #2>.

[15] B. Perens. "The emerging economic paradigm of open source". *First Monday, Vol. 10, No. 10, 2005*, <http://www.firstmonday.org/issues/special10_10/perens/index.html>.

[16] K. Pohl, G. Böckle, F. van der Linden. "*Software Product Line Engineering: Foundations, Principles, and Techniques*". Springer, 2005. ISBN-10: 3-540-24372-0.

[17] J. Sincero, H. Schirmeier, W. Schröder-Preikschat, O. Spinczyk. "Is The Linux Kernel a Software Product Line?". En *F. van der Linden, B. Lundell (Eds.), Proceedings on the Third International Workshop on open source Software and Product Lines: OSSPL07 Asia, September 10, 2007*, Kyoto, Japan <http://itea-cosi.org/modules/wikimod/index.php?page=OssPlas07>.

[18] A. Stellman, J. Greene. *Beautiful Teams*. O'Reily 2009 pp. 103-111. ISBN: 0596518021.

[19] J. Wesselius. "The Bazaar inside the Cathedral: Business Models for Internal Markets". *IEEE Software Vol. 25, No. 3*, May/June 2008 pp. 60-66.

Notas

[1] TPPT Nuestra experiencia en la industria europea del software se alinea con la vision de Perens [15] quien estima que quizás el 90% del software en cualquier empresa no es diferenciador"

[2] DVTk (Dicom Validation Toolkit), software libre que soporta el intercambio de imágenes médicas <http://www.dvtk.org/>.

[3] <http://www.softwareproductlines.com/resources/vendors.html>.

[4] <http://stylebase.tigris.org>.

[5] <http://subversion.tigris.org>.

[6] <http://semantic-mediawiki.org/wiki/Semantic_MediaWiki>.

[7] <http://www.dvtk.org/>.

[8] <http://subversion.tigris.org>.

[9] <http://www.sei.cmu.edu/productlines/framework.html>.

Abordar las necesidades de la industria en Software Libre

Jan Henrik Ziesing
Fraunhofer FOKUS, Centro de Competencia de Qualipso de Berlín (Alemania)

<jan.ziesing@focus.fraunhofer.de>

Traducción: Miguel Angel Tinte García (GSyC/LibreSoft, Universidad Rey Juan Carlos)

Resumen: *el software libre está, además de cosechando un gran éxito, atacando los viejos prejuicios y las tradicionales dificultades del mercado. Para profundizar en estos temas, Fraunhofer Fokus está fusionando los resultados del proyecto europeo Qualipso con su demostrado modelo teórico sobre interoperabilidad en el sector público, orientándolo a la industria de las tecnologías de la información. De esta manera, podrá transferir el éxito obtenido y los modelos de negocio teóricos de interoperabilidad al mundo del software libre en la región de Berlín capital.*

Palabras clave: *interoperabilidad, necesidades de la industria, Qualipso, software libre.*

Autor

Jan Henrik Ziesing trabaja como científico e investigador para el Instituto Fraunhofer en sistemas de comunicación abiertos en Berlín. Con una amplia experiencia en la transferencia de investigación aplicada, Jan se ha incorporado recientemente a las actividades del Centro de Competencia del proyecto Qualipso en Berlín.

El uso y desarrollo de software libre se han incrementado significativamente en los últimos años. Tomando como punto de partida la investigación y las actividades dirigidas por la comunidad, un nuevo segmento de negocio ha sido desarrollado recientemente alrededor del software libre. Mientras algunas de las compañías más importantes en el mundo del software libre (por ej. RedHat) hasta ahora se han convertido en los principales agentes del mercado e incluso algunas de las mayores compañías de software (por ej. IBM, Sun…) han desarrollado sus propios productos y estrategias en software libre, éste ha sido hasta el momento un punto estratégico especialmente para las pequeñas y medianas empresas. Debido a la participación de empresas orientadas a obtener beneficios económicos, podemos observar una profesionalización en los procesos de desarrollo y también en términos de mantenimiento de recursos y continuidad [1][2][3].

Aunque la profesionalización e industrialización del desarrollo de software libre es un fenómeno ya visible, existen aún prejuicios por parte de los usuarios y problemas para las empresas que impiden un uso aún más amplio y una mayor variedad de ofertas [4].

La agencia de innovación TSB Berlin Gmbh ha publicado recientemente un estudio sobre el sector del software libre en Berlín que ofrece una visión general de la situación actual del mercado alemán en la región de la capital, mostrando las debilidades y las necesidades de las empresas y los agentes activos, así como el potencial de la región [4].

Según el estudio, las principales razones para el uso del denominado Software Libre por parte de la industria de Berlín son las ventajas económicas, las preferencias personales y razones de calidad, que son las tres mencionadas por más del 60% de los agentes industriales y claramente superan otras motivaciones de organización y de imagen, que son sólo mencionadas por menos del 30% (ver **figura 1**).

Las dos limitaciones más importantes para el éxito del software libre en el mercado (ambas mencionadas por cerca del 60% de los agentes más importantes relacionados con el software libre en Berlín) son el insuficiente apoyo político y los prejuicios existentes respecto a la calidad y sostenibilidad del software libre (ver **figura 1**). Además, otros limitadores menos mencionados serían la dificultad de competir contra los agentes dominantes en el mercado, la falta de modelos de negocio y financiación (algo por encima del 40%), así como las cuestiones legales (casi el 25%).

El sector del software libre en Berlín se caracteriza, por un lado, por la alta demanda de soluciones desarrolladas con software libre y una fuerte relación entre software libre e I+D. Por otro lado, la industria del software libre en Berlín se encuentra muy dispersa y no suficientemente integrada en una red principal especializada en software libre [4].

Analizando estas fortalezas, debilidades, oportunidades y amenazas para el desarrollo de la industria del software libre en la región de Berlín capital es reseñable que las oportunidades más importantes para la industria coinciden con las principales debilidades y las mayores fortalezas con las amenazas (ver **figura 2**). Una estrategia razonable sería por lo tanto abordar las amenazas cuando sea posible y fortalecer la industria del software libre en particular cuando se encuentren las oportunidades.

Para la región de Berlín, este esfuerzo implica en especial abordar los prejuicios sobre la calidad, el fortalecimiento de la integración en la cadena de valor y el fomento de la transferencia de conocimientos de I+D para aplicaciones industriales.

Mejorar la percepción de la calidad requiere no sólo usar sofisticadas herramientas de desarrollo orientadas a la calidad y a los procesos, sino también realizar pruebas del producto y su evaluación. Las evaluaciones de calidad deberían hacerse visibles mediante el marketing y la comunicación. Un paso importante hacia la mejora de la transferencia de conocimientos y la integración en la cadena de valor podría lograrse con una mejor conexión en red en la región de Berlín capital, que hasta ahora no ha tenido una estrategia clara ni ha sido suficientemente establecida.

Estos pasos, necesarios para el fortalecimiento de la industria del software libre en la región de Berlín, coinciden con el enfoque del centro del proyecto europeo Qualipso, que recientemente ha establecido centros de competencia de software libre en Madrid (España), Sao Carlos (Brasil) y Berlín (Alemania) y tiene previsto establecer más adelante nuevos Centros de Competencia en Italia, China, Polonia y Francia.

Estos centros de competencia prestarán servicios derivados de los resultados del proyecto Qualipso, que están dedicados a mejorar la calidad del software libre en los diferentes aspectos. Estos comprenden las herramientas y las mejores prácticas para una profesionalización e industrialización del proceso de desarrollo, pruebas de productos y etiquetado, medidas de la interoperabilidad, y evaluación jurídica, así como certificación y formación.

El segundo objetivo de los Centros de Competencia Qualipso es construir redes regionales. Los Centros de Competencia están habilitados para conectarse a través de la Red

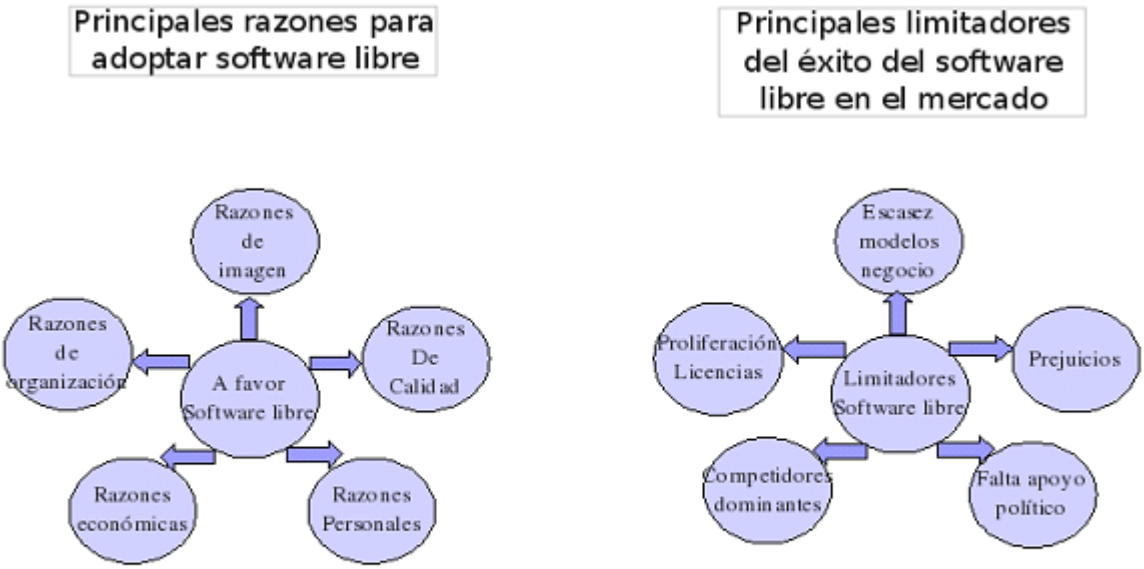

Figura 1. Principales razones y limitadores del software libre en la región de Berlín.

Qualipso, por lo que pueden ofrecer soporte a redes, apoyo y capacidad de intercambiar buenas prácticas no sólo en su región sino en todo el mundo.

*Fraunhofer FOKUS (*Fraunhofer Institute for Open Communication Systems*), como impulsora del Centro de Competencia de Qualipso en Berlín, ha adquirido una amplia experiencia en los últimos años en soluciones informáticas para los sectores públicos. Un tema importante ha sido y sigue siendo el paisaje heterogéneo existente en las tecnologías informáticas, lo que dificulta el intercambio de datos entre diferentes aplicaciones o entidades públicas.*

Las aplicaciones utilizadas en el sector público en Alemania están, en general, altamente especializadas para su aplicación en su dominio y para ofrecer todas las funciones y servicios para este dominio en una única herramienta. La interoperabilidad con otras aplicaciones del mismo o de un dominio diferente no se consideró importante. Para el intercambio de datos entre diferentes aplicaciones era bastante común imprimir un documento en

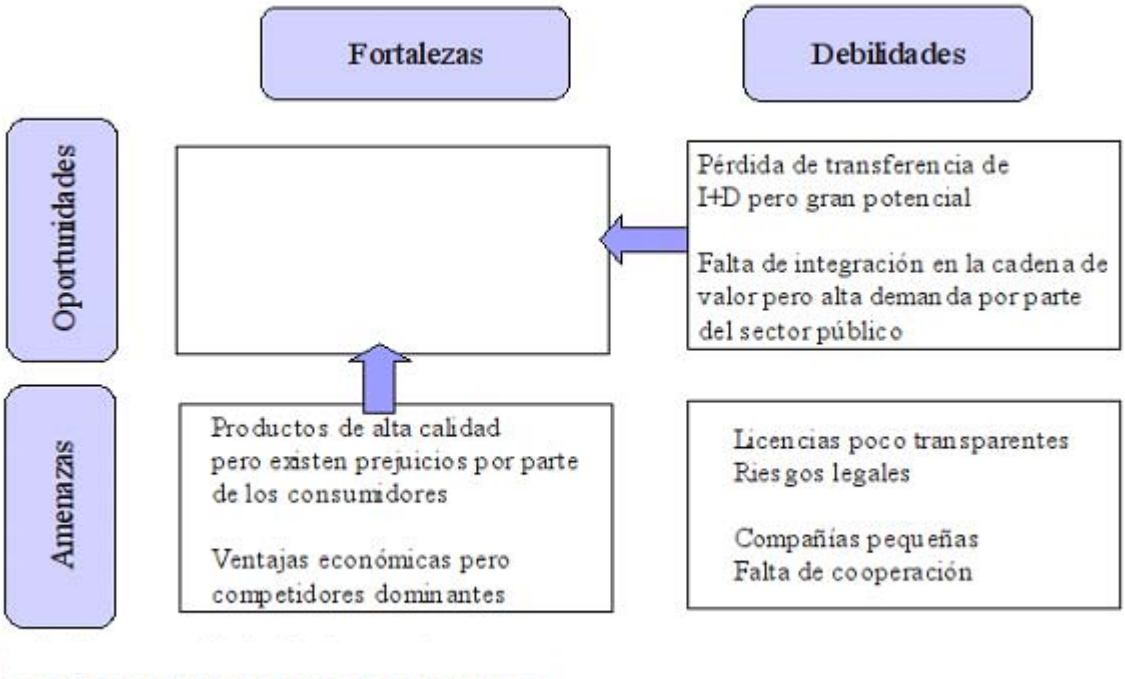

Figura 2. Análisis de la matriz DAFO.

una aplicación y escribirlo de nuevo a mano en otra aplicación.

Hoy en día, se hace evidente una tendencia hacia un eficiente y ergonómico proceso de administración electrónica basado en la conexión e integración de diferentes ámbitos de aplicación. Fraunhofer FOKUS se convirtió en uno de los principales impulsores de esta tendencia en Alemania mostrando en su Laboratorio de Administración Electrónica [5] las posibilidades y ventajas de una integración y orquestación de los diferentes dominios específicos de las aplicaciones software. El Laboratorio Administración Electrónica se dedica a investigar sobre interoperabilidad, a conectar a los principales actores de los sectores públicos y a comunicar sus logros.

Otra tendencia en la administración electrónica observada por Fraunhofer FOKUS es el desarrollo de servicios orientados a las arquitecturas en el sector público. Esto es especialmente interesante para la industria del software libre. La mayoría de las herramientas de software libre ofrecen determinados servicios para la solución de un cierto problema en numerosos ámbitos de aplicación, en lugar de ofrecer un conjunto de servicios integrados en una aplicación de un dominio específico. Mediante la transferencia por parte del Fraunhofer FOKUS de su interoperabilidad y sus conocimientos sobre el sector, el Centro de Competencia de Qualipso podrá hacer frente al mercado del software libre en el sector público mediante nuevos escenarios de interoperabilidad. Estos escenarios ayudarán a la industria a desarrollar soluciones específicas de dominio complejo mediante la oferta de mayores facilidades para la integración y orquestación de herramientas interoperables de código abierto.

Además de centrarse en la interoperabilidad de los servicios para software libre, el recientemente inaugurado Centro de Competencia de Qualipso en Berlín fomentará el desarrollo de software libre, ofreciendo una forja de última generación, así como la creación de redes de comunicación y oportunidades para la industria del software libre. El Centro de Competencia podrá, por lo tanto, hacer frente a las necesidades más urgentes de la industria del software libre en la región de Berlín, transfiriendo un enfoque previamente experimentado de Fraunhofer FOKUS al sector del software libre.

Referencias

[1] J.-P. Laisné at al. 2020 FLOSS Roadmap, 2008.

[2] D. Kühni. *Kurze betriebswirtschaftliche Analyse von Open Source Software*. (i. GmbH, Hrsg.) Bern, Schweiz, 2006.

[3] J. Lerner, J. Tirole. *Some simple economics of open source. Journal of Industrial Economics*, 50 (2), 2002.

[4] D.M. Fornefeld, M. Gasper. *Potenzialanalyse im Technologiefeld Open Source in der Hauptstadtregion Berlin.* (T. I. GmbH, Hrsg.) Berlin, Deutschland, February 2009.

[5] Fraunhofer FOKUS. Fraunhofer FOKUS eGovernment-Lab. <http://www.fokus.fraunhofer.de/en/fokus_testbeds/egov-lab/index.html>.

SpagoWorld, la iniciativa de software libre de Engineering

Gabriele Ruffatti
Director de Arquitecturas y Consultoría, Grupo Engineering I+D, Italia

<gabriele.ruffatti@eng.it>

Traducción: Israel Herraiz Tabernero (Universidad Complutense de Madrid)

Resumen: *este artículo muestra un caso de estudio basado en la experiencia de una empresa informática en el desarrollo y la gestión de proyectos de software libre. En la primera parte, presentamos la compañía y su papel en actividades relacionadas con software libre. Después, el artículo presenta las razones y la estrategia corporativa para entrar en el mundo del software libre, y el modelo de negocio de SpagoWorld, la principal iniciativa sobre software libre de Engineering. Finalmente, se investiga el doble papel que juega en actividades comerciales y creación y gestión de comunidades, con énfasis en la sostenibilidad del modelo.*

Palabras clave*: empresas informáticas, estrategias corporativas, software libre.*

Autor

Gabriele Ruffatti es director de la unidad de Arquitecturas y Consultoría en la división de I+D de Engineering. Con más de 27 de años de experiencia en el ámbito de la informática, ha ejercido varios puestos de gestión en diferentes áreas dentro del Grupo Engineering, como por ejemplo el desarrollo de productos complejos y de proyectos para importantes firmas y para el sector público, en procesos de mejora de calidad de software y de aseguramiento de calidad, además de en la definición de soluciones innovadoras. Además, contribuyó al inicio y desarrollo del sistema corporativo de calidad, y ayudó a conseguir las certificaciones CMMI e ISO 9001:2000. En 2004 lanzó la iniciativa de software libre SpagoWorld. Además, Gabriele coordina las actividades de software libre en Engineering. Es miembro del Consejo de SpagoWorld y del consorcio OW2. De 2006 a 2008, participó como profesor asociado, impartiendo clases sobre software libre, en el Departamento de Matemáticas, Física y Ciencias Naturales de la Universidad de Padua, dentro del programa de Máster en Informática.

1. Introducción

Engineering[1] es una empresa informática global, y el mayor operador informático de Italia en el sector servicios, proveedor líder de ofertas completas e integradas en toda la cadena de valor del software: diseño, desarrollo, servicios externalizados, productos y soluciones privativas verticales, consultoría estratégica e informática… ajustadas a los modelos de negocio de nuestros clientes en todos los mercados.

Hace ya más de cinco años desde que Engineering decidiera desarrollar y gestionar directamente proyectos de software libre, en lugar de simplemente colaborar con varias comunidades de software libre, o usar directamente sus resultados. SpagoWorld[2], la principal iniciativa de software libre de Engineering, constituye un ecosistema que reúne empresas, integradores, vendedores, instituciones y usuarios operando de una manera efectiva para añadir valor al dominio del software libre.

La construcción de un ecosistema de software libre efectivo requiere algunos aspectos clave:

- Colaboración con las comunidades de software libre a nivel internacional.
- Realización de proyectos efectivos en el ámbito de las tecnologías actuales más estratégicas, presentando al mismo tiempo una visión innovadora.
- Éxito comercial.

En la actualidad, SpagoWorld es un ecosistema de software libre basado en proyectos que se enfocan en inteligencia de negocios y dominios de gestión de procesos de negocios (SOA, Service-Oriented Architecture), los cuales han sido identificados por Engineering como los más prometedores en el mundo del software libre. Los proyectos se alojan en la comunidad global OW2[3] y añaden un enfoque original a las capacidades estándar de sus dominios específicos, que se caracteriza por una especial atención a las necesidades de los usuarios, siendo ésta la razón del incremento de su rendimiento comercial.

La iniciativa SpagoWorld no es sólo cooperación tecnológica; su objetivo principal es promocionar las comunidades de los proyectos, a la vez que se asegura la libertad del software desarrollado a tiempo y de las colaboraciones con empresas, de manera que se consolida un modelo de negocio diversificado y basado en un enfoque comercial del software libre. Los aspectos más destacados de este enfoque son: desarrollo de soluciones adaptables con énfasis en los requisitos de los usuarios, desarrollo de soluciones a nivel empresarial, soporte a los servicios ofrecidos, atención prestada a las demandas de las comunidades de software libre y la comunidad académica, y, finalmente, todo lo que concierne al crecimiento de un ecosistema que crea nuevo valor para todos sus participantes.

SpagoWorld es un ejemplo efectivo de ecosistema de negocios que actúa con un modelo de negocio específico, basado en el desarrollo y la promoción de soluciones particulares (mediante la venta de soporte a servicios y proyectos de software relacionados) y al mismo tiempo sostiene el sistema entero en un verdadero entorno de *coopetición*[4]. El dominio del software libre es particularmente apropiado para el desarrollo de una estrategia colectiva que incrementa el valor en un contexto que puede ser definido como "ecológico", donde las ganancias no monetarias y difícilmente cuantificables pueden ser más beneficiosas que las monetarias, porque son beneficios a largo plazo en un contexto de sostenibilidad.

2. Engineering y el software libre

Engineering lleva a cabo la definición de soluciones con arquitecturas innovadoras, y la realización de proyectos complejos para la administración publica, instituciones financieras y grandes empresas, usando tanto software libre como soluciones privativas.

Después de una primera fase centrada en el apoyo a iniciativas de software libre y la investigación de comunidades, desde el año 2004 Engineering ha definido una estrategia enfocada en el desarrollo, implementación y entrega de soluciones basadas en software libre como una oportunidad tecnológica y de negocio en los servicios que ofrece a sus clientes, siendo sus objetivos principales los siguientes:

- Uso industrial del software libre en las actividades de integración de sistemas, y provisión de un soporte completo a los servicios ofrecidos.
- Contribución activa en varios proyectos de software libre.
- Presencia activa en comunidades internacionales.
- Desarrollo y apoyo de iniciativas de software libre.

En la actualidad, la compañía cree que el software libre es un factor acelerador para la

consecución de objetivos estratégicos críticos en la empresa.

La compañía proporciona a sus clientes su experiencia y conocimiento para la selección, integración, validación y soporte de los mejores componentes de software libre, incluyendo sus propias soluciones, ofreciendoles los beneficios completos de un mayor ratio de valor monetario, gracias a atractivos esquemas de licenciamiento de software libre, y a la robustez como resultado de un intenso escrutinio del código fuente.

El enfoque sobre el software libre es racional desde un punto de vista ingenieril: analizar las ventajas e inconvenientes, y tomar decisiones adecuadas mejor que ideológicas. Engineering está convencida de que un uso adecuado del software libre puede ayudar a generar altos retornos para sus clientes, a la vez que se mejora la calidad de los sistemas.

3. El proceso de liberación de software de Engineering: razones y estrategia

En los últimos 30 años, Engineering ha operado en los mercados de desarrollo de software e integración de sistemas. Hace seis años, la compañía comenzó a pensar que el software libre era útil para los negocios. En aquella época, el software libre no estaba tan extendido en el mercado como lo está ahora, y la cuestión era si el software libre tenía el potencial para proporcionar beneficios en el mercado de integración de sistemas. Para responder a esta cuestión, la compañía analizó las características particulares del software libre, y determinó que, por varias razones, era posible elaborar un modelo de negocio válido para los integradores de sistemas, que podían aprovecharse de estas características.

Resumiendo, la completa ausencia de costes de licencia puede liberar recursos financieros que el cliente puede dedicar a la adquisición de servicios y de soluciones a medida, aspectos fundamentales ambos del negocio de un integrador de sistemas. La disponibilidad del código fuente permite al integrador de sistemas extender su oferta en las áreas de soporte y servicios al mantenimiento de soluciones de software libre. Este mercado potencial tiene una barrera de entrada muy baja, de modo que no existen posiciones dominantes.

Además, hay que tener en cuenta que la principal característica de un profesional informático, que tiene que encontrar la mejor solución para su propio cliente, está representada por su competencia adquirida gracias a la experiencia. Por tanto, es inevitable que la preparación en el dominio del software libre se haga *haciendo*, es decir mediante la gestión y participación activa en proyectos de software libre.

Orazio Viele, gerente de I+D en Engineering, escribió en 2005: *"El software libre representa más una oportunidad que una amenaza para un integrador de sistemas. En la actualidad, no podemos establecer cuál es el valor potencial de mercado estimulado por el software libre. Sin embargo, las características de este fenómeno son tales que podemos predecir un crecimiento progresivo en los próximos años. El reto para un integrador de sistemas consiste en una excelente preparación, porque ésta es la única manera de tomar ventaja de los mejores resultados provocados por esta revolución".* [1].

Tras examinar estas razones, se hizo necesario definir cómo enfocar esta oportunidad y definir una estrategia para posicionar con éxito a Engineering en este mercado.

Durante estos últimos seis años, Engineering ha mejorado esta estrategia, basándose en los siguientes principios:

- Tomar una posición líder en el mundo del software libre, desarrollando sus propias soluciones y creando un ecosistema alrededor de ellas. Esto diferencia a la compañía de otros integradores de sistemas, que sólo usan soluciones desarrolladas por otros. La iniciativa SpagoWorld es una acción concreta, para llevar a cabo esta estrategia. Todo esto hizo posible que Engineering fuera percibido como un productor de soluciones de software libre, especialmente fuera de Italia.
- Ser parte y contribuir a las comunidades internacionales de software libre, para crear una red colaborativa que enriquezca su oferta al mercado de soluciones y servicios. El ser miembro del consorcio OW2 y las contribuciones a diferentes comunidades son pruebas claras de esta estrategia.
- Seleccionar soluciones de software libre de alta fiabilidad, para ofrecerlas a sus clientes y satisfacer sus demandas. Dos herramientas adoptadas para alcanzar este objetivo son la creación de un centro de competencia dedicado en exclusiva a explorar soluciones, y la definición de una metodología para evaluar software libre.

La estrategia adoptada por Engineering respecto al software libre está centrada en dos aspectos más:

- Desarrollo de *software libre y gratuito*, teniendo en cuenta que Engineering no adopta un esquema de licenciamiento dual, sino que desarrolla y gestiona software liberado únicamente con licencias libres (como la licencia Apache o la GNU GPL/LGPL[5]), evitando cualquier política que asemeje el software libre al modelo privativo[6].
- Un modelo de negocio *centrado en el proyecto*, donde el proyecto es más importante que las soluciones tecnológicas que se adopten, y donde se explota la capacidad para diseñar y desarrollar sistemas complejos capaces de satisfacer las demandas de los clientes. En este modelo, el uso de software libre se convierte en una herramienta para mejorar las habilidades de diseño y el *know-how* técnico de un integrador de sistemas.

Esta estrategia no se planeó *a priori*, sino que deriva de un análisis de resultados, y de su crecimiento y adaptación en el tiempo. Así, tras algunos años de actividad, podemos decir que hemos vivido la experiencia de integrarnos en un entorno ecológico competitivo, donde el concepto estratégico ha cambiado respecto al previamente conocido. *"Para el desarrollo de una estrategia, normalmente es importante fijar objetivos a corto, medio y largo plazo, y definir un plan, llamado estratégico, a través del cual podamos alcanzar esos objetivos, definiendo los medios más adecuados, y suponiendo que siempre serán insuficientes. Adaptar un enfoque ecológico del valor supone tanto abandonar la parte de certeza que se deriva de la posesión de un plan definido para desarrollar un cierto conjunto de objetivos, como cambiarlos debido a nuestra capacidad de exploración continua de nuestra existencia en el mundo y al sentido que tenemos que asignarle a este hecho. El objetivo es mejorar la calidad de nuestra condición, experimentando todos los posibles potenciales distintos."* [2].

4. El modelo de negocio de software libre de Engineering

Engineering considera en su modelo de negocio que en la actualidad el software libre no es simplemente un modelo de desarrollo y distribución de software, sino que está correlacionado con la naturaleza de las comunicaciones, con su evolución en el tiempo junto a los modelos de negocio, y con su estructura ecológica (Véase [3] para una comparación entre ecosistemas de negocios y ecosistemas biológicos).

En la actualidad, Engineering es un agente informático global, aunque es principalmente un integrador de sistemas con características particulares, cuyos resultados se valoran en el mercado como los resultados de una empresa informática con rendimientos efectivos de desarrollo, comenzando por:

- La capacidad "genética" para el diseño.
- Un énfasis importante en la adquisición de conocimientos de primera mano (primero "testea", luego propón a los clientes y finalmente desarrolla), crecientes en el tiempo mediante experiencia y evaluación.
- Independencia de la solución elegida y adoptada.
- Flexibilidad con respecto a las diferentes situaciones, y una actitud natural para adaptar cada solución a distintos contextos, de modo que se generen soluciones y productos con un alto nivel de adaptación al cliente.
- Un enfoque industrial que hace énfasis en la adopción de los resultados en el mundo "real", de una manera efectiva.

Estas características también se aplican al caso del software libre, especialmente en el diseño y desarrollo de soluciones, con una actitud independiente y un enfoque en la utilidad industrial.

Como resultado, el modelo de software libre de Engineering es un modelo *empresarial de software libre*, en lo que respecta al desarrollo de proyectos y soluciones abiertas, y un modelo *profesional de software libre* en lo que respecta a la certificación de competencias y servicios de soporte a soluciones abiertas.

Más en detalle, el modelo de desarrollo de software libre puede calificarse como centrado en el proyecto, porque "*el alcance del desarrollo de una solución específica comprende la realización de proyectos de software bajo las demandas de diferentes clientes, que se pueden beneficiar de una solución de software libre, ofreciendo mejores características que una solución propietario en términos de disponibilidad, apertura, modificabilidad, modularidad, integración, adaptabilidad, reusabilidad y escalabilidad. Resumiendo, el proyecto tiene más valor que la solución adoptada*" [4].

Según este modelo, las soluciones basadas en software libre hacen posible el desarrollo de proyectos; estos proyectos, de hecho, posibilitan el crecimiento del software libre. Por otro lado, el software libre favorece la consecución de los requisitos de los clientes, de modo que se logra la mejor solución adaptada a sus necesidades; por otro lado también, la comunidad se beneficia de estos nuevos requisitos, nuevo código, testeo, realimentación y contribuciones externas.

En este contexto, el integrador de sistemas es el facilitador que activa las relaciones sinérgicas, y uno de los principales actores, gracias a su capacidad "genética" para trabajar de esta manera.

5. La iniciativa SpagoWorld: un poco de historia

La iniciativa SpagoWorld es un ejemplo de cómo Engineering dirige su modelo de negocio de software libre centrado en proyectos.

Surgida en 2004, la iniciativa cuenta ahora con cuatro proyectos principales:

- *SpagoBI*: la plataforma para *business intelligence* (BI).
- *Spagic*: la platforma empresarial de integración SOA/BPM.
- *Spago4Q*: especialización de SpagoBI para calidad de software.
- *Spago*: una plataforma empresarial Java.

Todos estos proyectos adoptan el mismo modelo de licenciamiento (se distribuyen bajo una licencia GNU LGPL, sin versiones "profesionales" o "*enterprise*" por las que es necesario pagar), y se alojan en el consorcio OW2, que proporciona un soporte independiente y a largo plazo gracias a una comunidad global.

Los proyectos comparten una visión común, basada en:

- *Soluciones flexibles*: integrando componentes ya existentes y desarrollando nuevos módulos, con un enfoque de integración en la plataforma, de modo que se identifican las soluciones más adecuadas para las necesidades de los usuarios.
- *Nivel empresarial*: las soluciones resultan de la experiencia en proyectos a nivel empresarial, donde las aplicaciones son críticas, y hay que garantizar la funcionalidad y un alto rendimiento.
- *Enfoque en el desarrollo de proyectos*: la capacidad para entender las necesidades reales de los usuarios, además de los requisitos de los diferentes proyectos, convierten a las soluciones de SpagoWorld en la mejor opción para comenzar el desarrollo de un nuevo proyecto de software.
- *Uso comercial:* las soluciones de software libre adoptan un esquema libre de licenciamiento, que permite el uso de las soluciones de SpagoWorld en diferentes categorías de productos y servicios.
- *Servicios de soporte*: cada solución es entregada, bajo demanda, incluyendo una solución completa de servicios de soporte.
- *Enfoque en la comunidad y la investigación*: todos los proyectos tienen un alto compromiso con las necesidades de la comunidad, e incorporan resultados innovadores de investigación.
- *Creación de un ecosistema de valor*: la iniciativa SpagoWorld participa en la creación de un ecosistema de valor añadido para todos los miembros de la comunidad: empresa, desarrolladores, investigadores y usuarios.

6. SpagoWorld y el mercado comercial de software libre

Las soluciones SpagoWorld son descargadas desde diferentes países alrededor del mundo. Esto confirma que son conocidas (y probablemente usadas[7]) en un contexto global, sin ceñirse a un mercado geográfico específico.

Aunque el éxito comercial significa otra cosa diferente. Este aspecto debe considerar dos puntos preliminares, uno particular a cada proyecto y otro que tiene que ver con la solución como un todo.

La premisa específica consiste en el hecho de que cada solución tiene que enfrentarse a competidores específicos en un contexto de software libre, donde la reputación otorgada por la comunidad es crucial (por ej., Spago, el *framework* Java, apenas puede competir con otras *frameworks* Java más conocidos; Spago4Q es una solución nueva y única, sin competidores reales, pero necesita alcanzar una reputación suficiente; Spagic, una solución bien recibida por la comunidad, necesita algo de tiempo para demostrar su eficacia en el uso en casos reales alrededor del mundo; SpagoBI, la más popular, es reconocida sobre todo como un producto software libre).

La premisa general se refiere a diferentes aspectos:

- Engineering, desarrollador de todas las soluciones, es un integrador de sistemas que actúa en los mercados y dominios informáticos, y que está siguiendo una trayectoria internacional todavía sin finalizar. Los competidores de software libre son empresas que actúan en un dominio específico (como Pentaho, Jaspersoft y Actuate en el dominio de *business intelligence*, o Intalio y MuleSource para soluciones SOA/BPM) ofreciendo soporte a nivel mundial. Contrariamente a un integrador de sistemas de propósito general, el mercado informático percibe a estas empresas como más eficaces centrándose en un único dominio que es crítico para su éxito, e invirtiendo cantidades significativas en actividades de marketing.
- Especialmente en Europa, los grandes integradores de sistemas que dirigen el mercado informático prefieren no usar las soluciones de otro integrador de manera explícita, para invadir "intrusismos";
- La falta de soporte constante y universal en regiones como Europa, EE.UU., Asia y Latinoamérica no facilita la inserción de las soluciones de SpagoWorld en las listas "*top*", a pesar de sus funcionales y sus innovadoras características.

No obstante, tanto SpagoBI como Spagic gozan de una buena reputación (su reciente inclusión en Gartner Research [5] [6] confirma este punto), y han obtenido pronto ciertos resultados comerciales, de modo que se han convertido en dos activos de software libre del grupo Engineering: Spagic está ganando impulso en el desarrollo de proyectos SOA/BPM para clientes de Engineering, algunos de ellos multinacionales; las actividades de desarrollo de SpagoBI se sostienen gracias a los beneficios que proporcionan las ventas de servicios de formación y soporte, principalmente en Francia y mercados relacionados. Además, los servicios de soporte de SpagoBI se ofrecen tanto en Latinoamérica como en Asia.

7. SpagoWorld y la comunidad de software libre: lecciones aprendidas

El primer resultado significativo de Engineering en el software libre proviene de sus actividades en algunos proyectos de investigación en Italia y Europa.

Las colaboraciones en este campo con em-

presas, universidades e institutos de investigación han proporcionado oportunidades para desarrollar soluciones y componentes libres, entre los que podemos mencionar bxModeller para el modelado de procesos de negocio, que proviene de los resultados de los proyectos de investigación DISCoRSO, X@Work y TEKNE; las soluciones del proyecto Bricks para bibliotecas de herencia cultural digitales; SeCSE para sistemas centrados en servicios y ETICS, para el desarrollo de software y calidad en entornos *grid*.

Cuando Engineering decidió comenzar a desarrollar y gestionar proyectos de software libre gratuitos a nivel industrial, se tomaron dos importantes decisiones:

- La elección de no hacerlo solos, sino conectados a una comunidad internacional ya existente.
- Ser consecuentes con el hecho de que en las comunidades de software libre los tiempos han cambiado.

De hecho, mientras que la naturaleza del software libre ha cambiado en el tiempo, desde el campo de las infraestructuras hacia el *middleware* primero, y hoy en día hacia un entorno aplicativo, se han creado nuevas comunidades con diferentes connotaciones (por ejemplo, las Fundaciones de Linux, Apache y Eclipse, el consorcio OW2, etc.).

Desde aquellas primeras comunidades de individuos movidos por la *ética hacker* nos hemos desplazado a la tercera generación de comunidades de software libre que aglutinan diferentes entidades legales, compañías federadas, vendedores, clientes, administraciones públicas e individuos. El modelo de negocio de estas meta-organizaciones es colectivo: el núcleo de su valor se orienta hacia el incremento del valor de la organización como un todo, y consiste en la estimulación de la colaboración entre sus miembros para alcanzar diversas metas que son útiles para todos los miembros.

La naturaleza cambiante de estas comunidades está influyendo también en el desarrollo de los actuales proyectos de investigación de Engineering. Algunos ejemplos de estas comunidades son:

- Qualipso[8], una alianza internacional de empresas informáticas, PYMEs, investigadores e instituciones públicas y académicas, cuyo rol consiste en ayudar a la industria y a los gobiernos en el fomento de la innovación y la competitividad con software libre.
- El Grupo de Trabajo sobre Software Libre de NESSI[9] que da soporte a la *Networked European Software and Services Initiative* para construir una plataforma tecnológica europea dedicada al software y a los servicios. Apoya a NESSI en la definición de una estrategia global sobre software libre, enfocada a empresas que quieran implementar o adoptar software libre, además de a comunidades de software libre que quieran colaborar y participar en NESSI, proporcionando el apoyo, la ayuda y la información necesarios para posicionar al software libre como el canal principal de comunicación de los logros de NESSI, asegurando niveles adecuados de calidad, dependencia y seguridad.

Incluso cuando Engineering estaba buscando una comunidad de referencia para sus proyectos de software libre, eligió una comunidad que prestaba atención especial a los requisitos de los usuarios finales y las empresas del ecosistema, sin ignorar el papel de los individuos y los desarrolladores en la comunidad. La adhesión al consorcio ObjectWeb en 2005 ha representado la elección particular de entrar en una comunidad caracterizada por una identidad precisa que persigue promocionar y afirmar un ecosistema sostenible y durable basado en soluciones software libre. Estas características pueden ayudar a las empresas del consorcio a conseguir sus objetivos de negocio y a las administraciones y a los usuarios a satisfacer sus demandas.

La participación en ObjectWeb también ha representado la oportunidad de compartir su transformación en el consorcio OW2, un nuevo consorcio internacional más fuerte, guiado por una filosofía abierta y que persigue la promoción de soluciones tecnológicas y una nueva manera de hacer negocios. Nacido en enero de 2007 como una consecuencia de la integración de las comunidades europea ObjectWeb y china OrientWare, OW2 es hoy un consorcio industrial independiente dedicado al fomento de un vibrante ecosistema de negocio, que cuenta con más de 100 organizaciones y 6.000 desarrolladores distribuidos en Europa, Asia y América, además de albergar más de cien proyectos tecnológicos.

La adhesión a OW2 ha sido una muestra de la estrategia de software libre de Engineering, cuyo objetivo es compartir proyectos con la comunidad e integrarlos con otras soluciones, buscando de manera continua nuevas oportunidades. El software SpagoWorld está alojado en la forja de OW2, de modo que permite a la comunidad participar usando herramientas específicas (listas de correo, foros, repositorios, área de descargas), y proporcionando una gestión independiente del software publicado. Esta colaboración va más allá de un apoyo notorio al crecimiento de toda la gama completa de software de OW2, y representa la participación en uno de los ejemplos más exitosos de comunidad de software libre de tercera generación.

En la actualidad, Engineering es co-fundador y miembro estratégico del consorcio OW2. Además, es particularmente activo en la vida del consorcio, siendo miembro del Consejo Rector, de todos los consejos de apoyo a los ecosistemas (Consejo del Ecosistema, Consejo Tecnológico y Oficina de Gestión), y permanece activo en muchas iniciativas (es el líder de la iniciativa de Business Intelligence, y forma parte de las actividades del Capítulo Local Europeo).

Con el tiempo, la colaboración se ha extendido también a otras comunidades. Gracias principalmente al desarrollo de la plataforma Spagic, Engineering participa en la comunidad Eclipse en el desarrollo del proyecto Eclipse STP/Intermediate Model, junto al instituto francés de investigación INRIA, y en la comunidad Apache como contribuidora al proyecto ServiceMix.

Engineering usa sus propias competencias en gestión de software libre, incluso usando soluciones software libre de terceros como componentes importantes de sus propios proyectos, o cuando necesita extenderlos, adaptarlos o resolver sus defectos de software. En este caso, Engineering proporciona las mejoras que realiza a la comunidad, obteniendo un doble beneficio:

- El soporte (de) y la participación activa en el ecosistema de software libre.
- La protección de la inversión del cliente para el que se ha realizado el proyecto. De hecho, si la contribución ofrecida es aceptada como parte de una nueva versión de la solución de software libre, hay claros beneficios en términos de mantenimiento y evolución futura.

Entre estas contribuciones, en el caso de desarrollos en Java, arquitecturas SOA y Business Intelligence, podemos nombrar los proyectos eXo Platform, Cimero, Jpivot, Harvard JHOVE, además de los mencionados previamente Eclipse STP y Apache ServiceMix.

La historia de las relaciones de Engineering con las comunidades muestra que:

- No sólo la *participación en la comunidad*, sino también la *"construcción de comunidad"* son cruciales, tanto en las comunidades ya existentes como en la creación de nuevas comunidades. Esto permite extender el "efecto red", facilitar la reputación global, intercambiar conocimiento y experiencias, y supone la aportación de contribuciones reales.
- La *gestión de la comunidad*, en la cual la empresa juega un papel líder en la gobernanza del proyecto, representa un elemento crítico y superable, especialmente cuando el proyecto presenta características especiales en términos de novedad y efectividad. Incluso en la experiencia de Engineering, es difícil construir una comunidad formada por actores que son activos en el soporte y la gestión del crecimiento del proyecto. El problema de las relaciones de confianza en la comunidad, entre la empresa y firmas de diversos tamaños, necesita todavía más investigación, puesto que es un

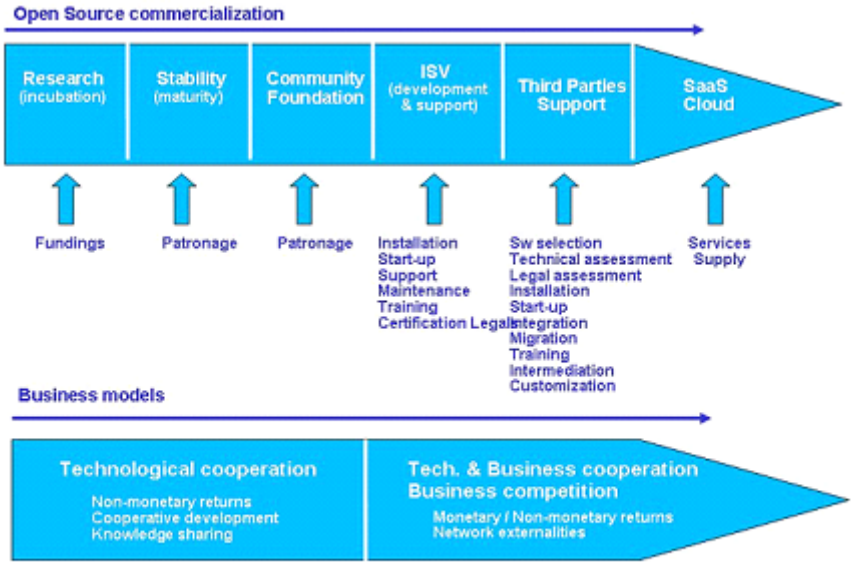

Figura 1. De la colaboración a la coopetición: Situando el software libre a nivel de negocio.

aspecto fundamental en la construcción de una verdadera ecología de valor.

■ *Las relaciones con las comunidades* son procesos de creación de valor respaldados por redes complejas de relaciones sociales y de negocio, de valores compartidos, y de interdependencias entre los diferentes miembros. El valor obtenido es alto, aunque principalmente no cuantitativo (por ej., no monetario). Es una colaboración en una red industrial basada en el conocimiento, que comparte esfuerzos promocionales y un fuerte compromiso en las decisiones cruciales, en el fomento de la innovación, en la competición abierta y en la libertad para formar parte de las diferentes actividades.

8. Sostenibilidad del modelo: una aproximación ecológica

La iniciativa SpagoWorld proporciona un contexto donde su comunidad y los diferentes actores (empresas, vendedores, integradores, consultores en BI, instituciones públicas, clientes, académicos e individuos) cooperan para desarrollar una infraestructura de soluciones madura y fiable, y compiten para conseguir sus objetivos, creando un entorno activo y estimulante. De este modo, todos aquellos que adopten SpagoWorld pueden encontrar un entorno de referencia para su estrategia de adquisición de software libre, y una oportunidad para contribuir al crecimiento de una estrategia colectiva que pretende incrementar valor en un contexto ecológico. La cooperación (en asuntos no-monetarios) y competición (en el mismo mercado) simultáneas posibilitan relaciones complejas que fomentan el ecosistema (ver **figura 1**).

Desde este punto de vista, la iniciativa SpagoWorld es un ejemplo efectivo de promoción de un ecosistema de negocios que actúa como el modelo de negocio específico de Engineering, y que se basa en el desarrollo y promoción de sus soluciones (vendiendo servicios de soporte y los proyectos de software correlacionados), y que al mismo tiempo apoya el sistema completo en un entorno de verdadera *coopetición*. En el contexto del software libre, los retornos indirectos y no monetarios pueden ser de más valor que los monetarios porque suponen beneficios en un contexto sostenible a largo plazo.

Referencias

[1] O. Viele. Software Libero, opportunità o minaccia per i system integrator?, *Homepage Engineering Ingegneria Informatica Anno VII n.8*, Roma 2005 (en italiano)

[2] A. Ganzaroli, L. Pilotti. En R. Fiocca "*Rileggere l'Impresa*" (Capitolo 15 «*Quali reti oltre il networking*»), ETAS, 2007 (en italiano).

[3] M. Iansity, R. Levien. Keystones and Dominators: Framing Operating and Technology Strategy in a Business Ecosystem. *Harvard Business School, Working Paper #03-061, 2004*.

[4] G. Ruffatti. Ecosistemi di business in azione. Il caso Engineering. *Quaderni di management – n. 33, E.G.V. Edizioni, May-June 2008*, pp. 37-48 (en italiano). Disponible también en: <http://www.spagoworld.org/ecm/faces/public/guest/home/community/resources>.

[5] A. Bitterer. *Who's Who in Open-Source Business Intelligence*. Gartner Research, abril 2008.

[6] M. Pezzini, L.F. Kenney, J. Lennard. *Cool Vendors in Platform and Integration Middleware*. Gartner Research, marzo 2008.

Notas

[1] Engineering <http://www.eng.it/>.

[2] SpagoWorld Initiative <http://www.spagoworld.org>.

[3] OW2 Consortium < http://www.ow2.org/>.

[4] El término *coopetición* significa la presencia contemporánea de relaciones de cooperación y competición.

[5] En la iniciativa SpagoWorld, que está basada en la estrategia comercial sobre software libre de Engineering, se requiere que el software se publique bajo la licencia GNU LGPL.

[6] El licenciamiento dual es un modelo híbrido, que incluye tanto una licencia libre (generalmente de la familia de licencias GNU) y su venta, o la venta de extensiones, bajo una licencia privativa siguiendo un esquema EULA (*End User License Agreement*). Algunos elementos de este modelo parecen más cercanos al modelo propietario que al del software libre. Por ejemplo, la presencia de inversores institucionales en las empresas que lo producen, el casi total control de la empresa que desarrolla la solución, y las características mismas de la solución, generalmente más típicas de un producto que de una plataforma.

[7] El número de descargas (esto es, el número de veces que un fichero es descargado por un usuario desde la web hasta su ordenador) se considera normalmente un número correlacionado con el éxito de un proyecto de software libre. Sin embargo, no tiene demasiado significado, porque existen muchos factores que pueden influir en este fenómeno, y tampoco dice mucho acerca del número de descargas que realmente se instalan y usan.

[8] Qualipso <http://www.qualipso.org/>.

[9] NESSI Open Source Working Group <http://www.nessi-europe.com/Nessi/WorkingGroups/AdoptionWorkingGroups/NESSIOpenSourceSoftwareWorkingGroup/tabid/269/Default.aspx>

Una oportunidad para las empresas de software libre: mercado emergente en los países en vías de desarrollo

Susana Muñoz Hernández, Jesús Martínez Mateo
Facultad de Informática, Universidad Politécnica de Madrid

<{susana,jmartinez}@fi.upm.es>

Resumen: *durante los últimos años el negocio del software ha cambiado en muchos aspectos. Las compañías de software han evolucionado desde producir aplicaciones software y vender su código propietario, a proporcionar aplicaciones de código abierto y centrar su negocio en ofrecer servicios relativos a su adaptación, instalación, mantenimiento y formación de usuarios. Las compañías de software libre emplean un modelo de negocio que proporciona nuevas oportunidades en mercados donde el software privativo es inviable. En este artículo describimos una interesante oportunidad de negocio que está surgiendo para estas compañías: las necesidades de software de los países en vías de desarrollo. Estos países están creando un mercado emergente para el desarrollo de software especialmente interesante porque si sus necesidades y limitaciones especiales son tenidas en cuenta son un escenario perfecto para el enfoque del software libre. Finalmente, será también discutida la influencia de algunos catalizadores que están realmente actuando en este mercado objetivo.*

Palabras clave: *mercados emergentes, oportunidades de negocio, países en vías de desarrollo, software libre.*

Autores

Susana Muñoz Hernández es Doctor en Informática por la Universidad Politécnica de Madrid (UPM), tiene un máster en Gestión de las Tecnologías de la Información por la universidad Ramón Llull de Barcelona y un grado en Relaciones Internacionales por la Sociedad de Estudios Internacionales de Madrid. Ganó el primer premio en el concurso nacional para jóvenes talentos de la Universidad de La Salle de Madrid en 2003. Tras varios años de experiencia profesional trabajando en empresas privadas, comenzó a trabajar como profesora asociada en la Escuela de Informática de la Universidad Politécnica de Madrid en 1998 donde lleva a cabo su actividad investigadora en el grupo BABEL. Es responsable de difusión y comunicación de la plataforma nacional española de software y servicios, INES. Ha establecido el grupo de cooperación TEDECO (*Tecnology for Development and Cooperation*) en 2006 y ejerce como directora desde entonces. Desde TEDECO, lidera la iniciativa Morfeo-CODE para la aplicación de software libre a la COoperación para el DEsarrollo en la comunidad Morfeo.

Jesús Martínez Mateo es estudiante de doctorado en Informática en la Universidad Politécnica de Madrid (UPM). Tiene un máster en Matemáticas de la Computación por la UPM. Desarrolla su actividad investigadora en el laboratorio de informática en la sección departamental de Análisis Numérico como integrante del grupo de investigación en Información Cuántica y Computación. Contribuye a varios proyectos, entre otros el proyecto Cenit Segur@ financiado por el Ministerio de Comercio e Industria de España para el estudio y diseño de una Red de Distribución Clave Cuántica Metropolitana. Compagina su trabajo investigador con el desarrollo de software libre (ej. Lan Core) y la participación en el grupo TEDECO (*Tecnology for Development and Cooperation*), siendo un miembro activo que ha participado en dos misiones de campo en Burundi.

1. El software libre en los mercados occidentales

¿Por qué luchar contra el mercado?

Es bien conocido que el software libre y de código abierto[1] tiene que ganar una dura batalla contra el software propietario para encontrar un espacio en el mercado occidental. Algunos de estos problemas, quizá todos ellos, son específicos de este mercado donde el negocio del software tiene una historia consolidada. Por otra parte, existe un escenario donde puede ser evitada esta cultura tribal; en los países en vías de desarrollo muchos de estos problemas simplemente no existen. En general, el desarrollo y el despliegue del software en los países en vías de desarrollo tienen una corta tradición. Esto es una considerable ventaja para la expansión del software libre en aquellos mercados con una concepción más moderna en relación con el software [1].

La bondad de los mercados

Existen varios casos en los que una nueva tecnología ha tenido menos problemas para establecerse en un mercado emergente que en los ya consolidados. Por ejemplo, a finales del siglo diecinueve era más sencillo abrir el mercado a las locomotoras eléctricas en aquellos lugares donde los trenes de vapor no eran aún habituales. Esto es similar, hoy día, donde muchos lugares aislados de países en vías de desarrollo han comenzado a utilizar directamente comunicaciones móviles sin haber tenido cualquier infraestructura previa de telefonía fija. En estos países se ha producido una brecha tecnológica en algunas generaciones en relación con la revolución industrial, el hardware, o las comunicaciones entre otras; entonces… ¿por qué no también con el software? Aunque quizá una pregunta más apropiada sería: ¿puede ser el software libre la tecnología del futuro en el desarrollo de software? Probablemente sí, en cuyo caso la historia podría volver a repetirse de una forma similar para ignorar el software propietario en beneficio del software libre en los países en vías de desarrollo. Bajo este supuesto es posible que los países en vías de desarrollo se conviertan en un serio candidato para adoptar una estrategia nacional a nivel de gobierno en relación con el software libre, sin pasar por la fase previa de la era del software privativo. Actualmente, siguiendo esta sospecha, algunas comunidades de software libre [2] se han dado cuenta de que existe un crecimiento y una oportunidad de negocio para ellos en estos países.

¿Tradición o PCI (Progreso, Cambio e Innovación)?

El problema más importante del software libre en las grandes compañías es que éstas ya tienen la mayoría de sus aplicaciones desarrolladas utilizando software propietario. La tradición es una poderosa desventaja puesto que las migraciones de software pueden implicar inversiones a corto plazo y las compañías raramente confían en las ventajas que a medio y largo plazo les puede suponer una migración a software libre, u otros productos. Por el contrario, encontramos algunas instituciones, organizaciones y compañías en los países en vías de desarrollo que están comenzando a generalizar el uso de aplicaciones software [3]. En estos casos, y por razones evidentes al tratarse de países en vías de desarrollo, el presupuesto para la compra de software es considerablemente reducido, de tal forma que la adquisición de software libre no es una opción sino una necesidad. En cualquier caso, el software libre es mucho más económico que la compra de una aplicación propietaria convencional, por lo que las desventajas tradicionales son oportunidades en los mercados de software de los países en vías de desarrollo.

Comercio justo

El comercio justo es un movimiento social organizado y un enfoque del mercado que tiene como objetivo ayudar a los productores en los países en vías de desarrollo al mismo tiempo que promueven la sostenibilidad. El comercio justo no es tan sólo prestigioso desde el punto de vista moral, sino también por su rentabilidad. En 2007, las mercancías etiquetadas como comercio justo tuvieron un nivel de ventas por valor de aproximadamente 2,3 billones de euros en todo el mundo (3,62 billones de dólares americanos), con un incremento del 47% de un año para otro.

Ha existido un debate en un foro especializado acerca de las similitudes entre el software libre y el comercio justo [4]. El origen de esta discusión fue una presentación de Matthew Edmonson del proyecto gubernamental *Open IT Up* que dio en el Reino Unido unas conferencias para proveedores de asistencia tecnológica[2]. En realidad, existen bastantes semejanzas teniendo en cuenta que ambos implican una elección y una preocupación por las personas. James Davis dice que como usuarios de software libre esperamos que se nos conceda la libertad de hacer lo que nosotros queramos con nuestro software, siempre y cuando no se restrinja lo que los demás pueden hacer con nuestro software modificado.

A pesar de que no estemos de acuerdo con esta declaración (puesto que podemos encontrar algunas diferencias sustanciales entre el software libre y el comercio justo [5]) este es un punto muy positivo para el software libre, el ser comparado en un debate público con tan prestigiosa empresa como es el comercio justo.

2. Catálisis del software libre en los países en vías de desarrollo

Las TIC en los países en vías de desarrollo

Lejos del ancestral concepto de "cooperación", mucho más cercano a la caridad, en los últimos cinco años los proyectos de cooperación están orientados al desarrollo de los países receptores. Aparte de los campos tradicionales de cooperación (salud pública, educación, o el suministro de alimentos entre otros) existen nuevas áreas donde la cooperación está jugando un rol decisivo, y las relacionadas con las Tecnologías de la Información y la Comunicación (TIC) son probablemente las más importantes. De hecho, la cooperación en TIC está directamente aplicada al resto de áreas de cooperación tradicionales.

Software libre para el fortalecimiento institucional

La mayoría de los proyectos de cooperación que incluyen desarrollo de software han optado por el uso de software libre. Esto es muy interesante si tenemos en cuenta que este software va a ser parte de muchas instituciones clave (Administración o Educación entre otras) de los países en vías de desarrollo [6]. El e-Gobierno está siendo especialmente promovido y muchos gobiernos de países en vías de desarrollo están apostando por el software libre siguiendo el consejo de consultores externos independientes y Organizaciones No Gubernamentales (ONG) [7][8]. Es evidente la influencia que este hecho tendrá en el desarrollo de software en el resto de los sectores de estos países.

Siguiente generación de programadores de primera categoría

En la actualidad, sin otra opción, las tendencias actuales en software libre y propietario en los países en vías de desarrollo vienen de países extranjeros. Sin embargo, se espera que en un futuro cercano sus primeras generaciones de ingenieros informáticos sean los responsables del desarrollo, pero sobre todo del mantenimiento y uso del software. Este es otro punto a favor del software libre. Los esfuerzos en educación están concentrados en el campo del *e-learning* y el desarrollo de la educación superior. La influencia del software libre en la educación de las nuevas generaciones de programadores y profesionales de los países en vías de desarrollo está siendo reforzada por el trabajo de las ONGs y la cooperación de las universidades occidentales [9].

3. Países en vías de desarrollo: mercados de software libre emergentes

El software libre es una alternativa consolidada para el desarrollo de software profesional que es altamente recomendada para los países en vías de desarrollo. Durante los últimos años existen estudios y esfuerzos desde estos países que apuestan por el software abierto y los modelos de software libre [10]. Unos quince países africanos ya han publicado recomendaciones de software libre frente a soluciones de software propietario (por ej, Angola, Benin, Kenia, Senegal, Sudáfrica, Tanzania, Uganda y Zambia entre otros). Esto puede ser aplicado a muchas áreas pero es especialmente importante para tres: Educación, enfocado principalmente en las bibliotecas digitales [11], desarrolladores de software, o soporte TIC para la educación general; Administraciones Públicas, con especial interés en el despliegue del e-gobierno; y Salud, no sólo proveyendo software para instituciones de salud sino también para desarrollar el crecimiento de la e-Salud y su utilización en áreas aisladas [12].

Aparte de estas áreas en las que la introducción del software libre ya ha comenzado, el sector privado está dispuesto a desarrollar un

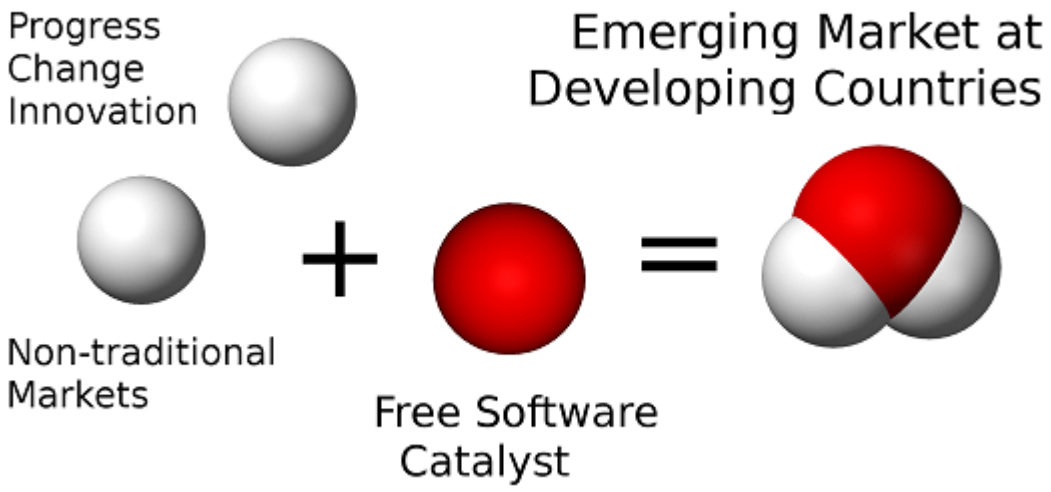

Figura 1. Proceso de catálisis del software libre en los países en vías de desarrollo.

software de apoyo confiable y seguro, y sus decisiones al elegir el tipo de software que van a utilizar está muy condicionada por el criterio de la Administración Pública y los sectores públicos en general.

Teniendo en cuenta las características especiales de los países en vías de desarrollo, su falta de reticencias a la hora de usar el software libre, el efecto catalizador de la cooperación y el previsible crecimiento de la industria de software en estos países (ver **figura 1**), podemos concluir diciendo que la aparición del mercado de software en los países en vías de desarrollo representa una oportunidad de negocio para las empresas de software libre.

Referencias

[1] J. Lerner, J. Tirole. "The Simple Economics of Open Source". *National Bureau of Economic Research, Inc., NBER Working Papers, No. 7600, 2000*. <http://ideas.repec.org/s/nbr/nberwo.html>.

[2] Morfeo-code. Iniciativa para el uso de software libre en la cooperación para el desarrollo, <http://tedeco.morfeo-project.org/>.

[3] G. Camara, F. Fonseca. "Information Policies and Open Source Software in Developing Countries". *Journal of the American Society for Information Science and Technology, 2007*.

[4] D. Wilcox. *Is Open Source Software Fair Trade for Nonprofits?* Designing for Civil Society. <http://partnerships.typepad.com/civic/2007/02/is_open_source_.html>.

[5] J. Davis. *How Free Software is not Fair Trade for non-profits*. Free web hosting for UK charities and non-profit organizations. <http://www.freecharity.org.uk/2007/02/08/how-free-software-is-not-fair-trade-for-non-profits/>.

[6] S. Weber. "*Open Source Software in Developing Economies*". Social Science Research Council, USA. <http://www.ssrc.org/programs/itic/publications/ITST_materials/webernote2.pdf>.

[7] United Nations Economic y Social Commission for Asia and the Pacific (ESCAP). "*ICT Policy database*," <http://www.unescap.org/icstd/policy/db/>.

[8] Programa para el Desarrollo de las Naciones Unidas (United Nations Development Programme, UNDP) y el Centro Internacional de Investigación para el Desarrollo (International Development Research Centre, IDRC) de Canada. *International Open Source Network* (IOSN) – Software Freedom For All, <http://www.iosn.net/>.

[9] TEDECO: Tecnología para el Desarrollo y la Cooperación. <http://tedeco.fi.upm.es/>.

[10] Bridges.org, CIPESA. *Free/Open Source Software (FOSS) policy in Africa: A toolkit for policy-makers and practitioners*, <http://www.cipesa.org/files/FOSSPolicyToolkit.pdf>.

[11] I. Witten, M. Loots, M. Trujillo, D. Bainbridge. "The Promise of Digital Libraries in Developing Countries" *Communications of the ACM, 44 (5)*, pp. 82-85, 2001.

[12] Fundación EHAS: Enlace Hispano Americano de Salud. <http://www.ehas.org/>.

Fecha de revisión de los enlaces web: mayo de 2009.

Notas

[1] En este artículo, hemos decidido utilizar el término *software libre* siguiendo la definición de la Free Software Foundation (FSF) en lugar de otros términos comunes, y seguramente más populares, como FOSS, del inglés *Free and Open Source Software*, o OSS, del inglés *Open Source Software*.

[2] **Nota del editor:** En la referencia aportada por los autores se comenta que Matthew Edmonson dio una conferencia a "circuit riders" que dan soporte tecnológico a empresas sin ánimo de lucro. Este término (también se puede usar "eRiders" en su lugar), tiene su origen en el movimiento metodista y se refiere a proveedores de asistencia tecnológica que atienden a pequeñas organizaciones sin ánimo de lucro de sectores económicos concretos para la resolución de problemas o el soporte a necesidades tecnológicas particulares de dichas organizaciones <http://en.wikipedia.org/wiki/Circuit_rider_(technology)>.