*Asociación para la Difusión y el Avance del Software Libre de Andalucía (ADALA)*
*Campogiblaltareños Entusiastas del Software Libre (CAGESOL)*
*Oficina del Software Libre. Universidad de Cádiz (OSLUCA)*

# Actas de las IV Jornadas Andaluzas de Software Libre

*Algeciras, 5 y 6 de noviembre de 2004*

**Organizadas por:**
ADALA
CAGESOL
OSLUCA
**Con la colaboración de:**
Escuela Politécnica Superior de Algeciras

**Editores:**
Alfonso Cepeda Caballos
Rafael Rodríguez Galván
Israel Herraiz Tabernero

*http://jornadas.adala.org*

Los textos que aparecen en el presente volumen constituyen las contribuciones presentadas en las IVJornadas Andaluzas de Software Libre. Reflejan la opinión de sus autores y se publican tal y como ellos las presentaron. Esta publicación no significa que ADALA, CAGESOL, OSLUCA ni los editores, compartan o apoyen las opiniones expresadas en dichos textos. Las contribuciones fueron propuestas por sus autores y pasaron un proceso de selección, durante el cual fueron revisadas por un comité de programa.

Por favor, use el siguiente formato para citar material de este libro:

Autor(es), "Título del artículo", en *Actas de las IVJornadas Andaluzas de Software Libre*, A. Cepeda, R. Rodríguez, I. Herraiz, Editores, números de página (2004).

ISBN: 84-88783-71-X

Publicado por:
**ADALA, Asociación para la Difusión y el Avance del Software Libre de Andalucía**
*http://www.adala.org*
**CAGESOL, Campogibraltareños entusiastas del software libre**
*http://www.cagesol.org*
**OSLUCA, Oficina de Software Libre. Universidad de Cádiz**
*http://www.softwarelibre.uca.es*
**Biblioteca de la Escuela Superior de Ingenieros de Sevilla**
*http://www.esi.us.es/BIB*


Impreso en España.

Tanto los trabajos individuales como las actas completas se pueden descargar una vez finalizadas las jornadas, en la página web *http://www.adala.org/jasl4/*.

*En memoria de* Manuel Estrada (Ranty) *y* Andrés García Solís (ErConde)
*que fallecieron en un accidente de tráfico al fomentar el software libre.*

# Índice General



## Artículo invitado



## Artículos presentados

## Organización de las jornadas

- Carlos Alberto Paramio Danta, codirector.
- Rafael Rodríguez Galván, codirector.
- Alfonso Cepeda Caballos, codirector.
- Israel Herraiz Tabernero, secretario.

## Comité de programa

- Gerardo Aburruzaga García, Dto. Lenguajes y Sistemas Informáticos. Universidad de Cádiz.
- Antonio Arauzo Azofra, Dto. Ingeniería Rural. Universidad de Córdoba.
- Alfonso Cepeda Caballos, Dto. Ingeniería de Sistemas y Automática. Universidad de Sevilla.
- José Manuel Frías Carnero, Dto. Lenguajes y Sistemas Informáticos. Universidad de Cádiz.
- Israel Herraiz Tabernero, Grupo de Sistemas y Comunicaciones. Universidad Rey Juan Carlos.
- Isidro Lloret Galiana, Dto. Lenguajes y Sistemas Informáticos. Universidad de Cádiz.
- Antonio Luque Estepa, Dto. Ingeniería Electónica. Universidad de Sevilla.
- Inmaculada Medina Bulo, Dto. Lenguajes y Sistemas Informáticos. Universidad de Cádiz.
- Manuel Palomo Duarte, Dto. Lenguajes y Sistemas Informáticos. Universidad de Cádiz.
- Francisco Palomo Lozano, Dto. Lenguajes y Sistemas Informáticos. Universidad de Cádiz.
- Carlos Alberto Paramio Danta, presidente de Campogibraltareños Entusiastas del Software Libre (CAGESOL)
- Ignacio José Turias Domínguez, director de la Escuela Politécnica Superior. Dto. Lenguajes y Sistemas Informáticos. Universidad de Cádiz.

## Programa

| Viernes 5 de Noviembre | | |
|---|---|---|
| Hora | Salón de Actos | Sala de Postgrado (junto a biblioteca) |
| 9:30-10:00 | **Inauguración** | |
| 10:00-10:30 | **Presentación de la distribución de la Universidad de Cádiz** | |
| 11:00-11:40 | **Proyectos fin de carrera y Software Libre**<br>Antonio Arauzo Azofra, Antonio Cubero y Lorenzo Salas | **Un proyecto de gestión integral de tiendas usando código abierto**<br>Jose Manuel Gonzalez Vida |
| 12:00-13:00 | **Gualadinex y aledaños**<br>Juan Conde, Consejería de la Innovación, Ciencia y Empresa | |
| 13:30-14:30 | **¿Qué es el Software Libre?**<br>Jesús González Barahona, Universidad Rey Juan Carlos | |
| 14:30-16:00 | Descanso para Comer | |
| 16:00-16:40 | **CVSAnalY: una herramienta libre para el análisis de repositos CVS**<br>Israel Herraiz, Gregorio Robles y Jesús M. González Barahona | |
| 17:00-17:40 | **wxPython: Aplicaciones multiplataforma con Python**<br>Arturo Fernández Montoro | |
| 18:00-18:40 | **DamFlow: un software de visualización de Flujos Hidrodinámicos desarrollado con Python, C++ y VTK**<br>Ana María Ferreiro Ferreiro, José Antonio García Rodríguez | **Aproximación al nivel de seguridad de soluciones basadas en Software Libre**<br>Alvaro López Ortega |
| 19:00-20:00 | **Últimos pasos de la Oficina del Software Libre de la Universidad de Cádiz**<br>Rafael Rodríguez Galván, director de la OSLUCA | |

| Sábado 6 de Noviembre | | |
|---|---|---|
| Hora | Salón de Actos | Sala de Postgrado (junto a biblioteca) |
| 10:00-11:00 | **KDE**<br>Antonio Larrosa, desarrollador de KDE | **SMB Web Client, Abriendo Windows a la WWW**<br>Victor M. Varela |
| 11:30-12:10 | **Cómo colaborar en el proyecto de software libre KDE**<br>Pedro Jurado Maqueda | **Taller de reutilización de PC's mediante PXES**<br>Círculos de Innovación y Tecnología. (aula 2.1 ,segunda planta; aula de informática) |
| 12:30-13:10 | **Gentoo Linux: filosofía e introducción**<br>José Alberto Suárez | |

## Invitados

- Juan Conde, Consejería de la Innovación, Ciencia y Empresa
- Jesús González Barahona, Universidad Rey Juan Carlos
- Rafael Rodríguez Galván, director de la OSLUCA
- Antonio Larrosa, desarrollador de KDE

Nota: La ponencia titulada Gentoo Linux: filosofía e introducción impartida por José Alberto Suárez se realiza aquí al no haberse podido exponer en las III Jornadas Andaluzas de Software Libre. El texto de la ponencia se puede encontrar en las actas de las III JASL que están disponibles en la página web *http://www.adala.org/jasl3/*.

# Presentación

El Software Libre se está popularizando. Y lo está haciendo gracias a iniciativas como, por ejemplo, las de las Juntas de Extremadura y Andalucía, donde los alumnos de los centros equipados con aulas informatizadas comienzan a trabajar con Linux y programas de código abierto; o como la de los usuarios del navegador Firefox, algunos de los cuales donaron una pequeña cantidad de dinero para, uniendo lo recaudado, publicar el anuncio de la salida de la versión 1.0 en el periódico The New York Times a toda página; o como la del famoso turista del espacio Mark Shuttleworth, que con el lema Ubuntu (o también "Humanity, caring and harmony") pretende llevar Linux y el Software Libre a toda la comunidad de usuarios de ordenadores del mundo. Todos ellos son proyectos destinados a dar a conocer al gran público que las herramientas libres que podamos necesitar para nuestro trabajo diario empiezan a estar más que preparadas para su uso cotidiano, por el profano o por el experto, sea cual fuere su edad o tarea a desempeñar. Una realidad que cada vez es más tangible, y que con el esfuerzo de todos, está generando un efecto de bola de nieve cada vez más difícil de parar.

Demos la bienvenida a la popularización del Software Libre, e invitémosla a quedarse. Recibamos pues con agrado a la libertad de uso, de innovación, y del conocimiento. Y hagamos de estas jornadas un punto de encuentro común donde la tecnología y la información esté al alcance de todo el mundo. Espero que todos disfrutemos con ellas tanto como la ilusión que hemos puesto la organización en prepararlas.

*Carlos Alberto Paramio Danta*
*Presidente de Campogibraltareños Entusiastas del Software Libre*

Lo que empezó siendo sueño de un grupo de hackers, desde un origen revolucionario basado en la defensa del conocimiento humano, entendido como aplicación al ámbito de las tecnologías de la información del modelo científico que ha llevado al hombre a la luna y al fondo del océano, hace ya muchos años que el software libre ha dado el salto definitivo y se ha situado como una perfecta opción para su uso por el gran público.

Sigiloso e insospechado, como un sueño, se ha ido adentrando desde ambientes favorables, universidades y centros de investigación, hacia nichos que hace unos años se dirían poco propicios: grandes compañías de ámbito tecnológico de la talla de Novell o IBM, administraciones públicas, como el gobierno de Brasil o las Juntas de Extremadura y Andalucía, nuestra región, donde los niños en las escuelas, todavía sin saberlo, dibujan muñecos usando programas de código libre y abierto.

Y si ha llegado tan lejos, lo ha hecho para quedarse entre nosotros. Porque no es un asunto de precio, sino de libertad. Porque cuando empiezas a valorar más la generosidad que la mezquindad, cuando empiezas a disfrutar de valores como el conocimiento y la solidaridad, cuando el compartir deja de ser un delito para convertirse en una rutina, en ese momento, ya no hay vuelta atrás. En ese momento, el software privativo deja de ser una alternativa, pues el resto de las consideraciones, técnicas, lúdicas, ergonómicas, pasan a un segundo lugar si no están unidas a la libertad.

*J. Rafael Rodríguez Galván*
*Director de la Oficina de Software Libre de la Universidad de Cádiz*

Durante estos últimos años las Jornadas Andaluzas de Software Libre se están consolidandos como una de las citas obligadas para toda la comunidad de software libre de Andalucía.

En este tiempo, el software libre ha avanzado profundamente junto con la evolución de internet. La mayor parte de las páginas web servidas utilizan como base un programa libre, se ha abierto la guerra de los navegadores al introducirse un navegador libre cuyo número de usuarios crece cada día, hay una alternativa libre para casi todos los programas privativos. Por otro lado, la mayor parte de las empresas y administraciones de todos los ámbitos se plantean la introducción de programas libres paulatinamente, unas veces motivado por la disminución de costes y otras por motivos estratégicos para no estar atado a una empresa específica.

Por todo esto, hemos elegido como tema principal de esta cuarta edición “Software Libre para todos”. El software libre ha pasado por varias fases, al principio proliferaban los programas con función de servidor o en linea de comando que no eran muy intuitivos. Sin embargo, en los últimos tiempos se ha incrementado el número de programas gráficos, con un entorno muy amigable que los hacen aptos para todo el mundo, con cualquier nivel de conocimiento de informática.

Estas jornadas pretenden fomentar el uso de programas libres, démosles una oportunidad y usémoslos.

*Alfonso Cepeda Caballos.*
*Consejo de dirección de ADALA*

# Avances recientes de la OSLUCA en el marco del Software libre en la Universidad Andaluza

J. Rafael Rodríguez Galván

***Resumen*— Se analizan en la primera parte de este documento las funciones de la Universidad Pública, enfatizando su compromiso social y ético además de sus fines formativo e investigador. Seguidamente, se detalla las estrechas relaciones que unen al software libre con cada una de estas funciones. Las conclusiones obtenidas justifican las dos iniciativas realizadas en los últimos meses por la OSLUCA (Oficina de Software Libre de la UCA): la propuesta y aprobación en Consejo de Gobierno de una normativa de intercambio de información institucional mediante estándares abiertos y la creación del sistema operativo** Guadalinex_UCA**, basado en** Guadalinex **y por tanto en** Debian GNU/Linux**.**

## I. Introducción

El pasado 15 de marzo de 2004 fue aprobada por parte del Consejo de Gobierno de la Universidad de Cádiz (órgano máximo encargado de establecer sus directrices estratégicas y programáticas), una declaración institucional de apoyo al software libre[1] que marca un importante hito, situando a nuestra universidad como abanderada en la adopción de estrategias de la información basadas en software libre en nuestra comunidad autónoma, codo a codo con la Junta de Andalucia y con un puñado de universidades e instituciones públicas españolas.

Esta declaración institucional estima que el fomentar el uso de Software Libre, entendiendo como tal a todo tipo de aplicaciones y recursos informáticos que puedan usarse, estudiarse, modificarse y distribuirse libremente, es una vía adecuada para:

- Potenciar el compromiso de la Universidad con la sociedad que la rodea, incentivando el desarrollo de la economía regional.
- Establecer vías de colaboración con otras universidades, instituciones y empresas de nuestro entorno.
- Fomentar la independencia tecnológica, racionalizar gastos y hacer de la UCA referencia en nuestro entorno para el avance de la Sociedad de la Información y el Conocimiento.
- Promover la educación en aspectos éticos positivos como la solidaridad, la colaboración y la legalidad.

Director de la OSLUCA (Oficina de Software Libre de la Universidad de Cádiz)

ISBN 84-88783-71-X©2004

Y en consecuencia, decide la creación de la Oficina de Software Libre de la Universidad de Cádiz (OSLUCA), encuadrada dentro de la infraestructura del Centro Integrado de Tecnologías de la Información (CITI), con el cometido de "promover el uso de las aplicaciones y recursos informáticos basados en software libre en la comunidad universitaria" y, en concreto, tomar medidas conducentes a:

- Garantizar la no discriminación de los usuarios de aplicaciones y sistemas operativos libres frente a usuarios de sistemas comerciales privativos
- Fomentar el uso y desarrollo de software libre en los miembros de la comunidad universitaria: profesores, personal de administración y servicios y alumnos. En particular, realizar tareas de formación, impulsar el software libre en las aulas informáticas, promover la publicación de material con licencia libre, utilizar herramientas libres para la docencia e investigación, así como para uso del personal de administración.

En el marco de sus atribuciones, superada una etapa de definición de objetivos y estrategias, la OSLUCA ha comenzado a producir en los últimos meses (e incluso en las últimas semanas) la algunas medidas encaminadas a plasmar el contenido de esta declaración insitucional. De éstas, tienen especial relevancia las dos que han sido aprobadas recientemente: En primer lugar, el decreto de intercambio de información institucional, también aprobado por el Consejo de Gobierno de nuestra universidad, que recoge la obligación a utilizar formatos abiertos por parte de sus miembros a la hora del intercambio de documentos. Y en segundo lugar, la publicación de Guadalinex_UCA, una adaptación del sistema operativo Guadalinex, basado de Debian GNU/Linux, fruto de la colaboración entre la OSLUCA y la Junta de Andalucía, con el fin de facilitar el acercamiento al software libre a las personas que forman parte de nuestra Universidad.

En las siguientes secciones, se analizarán con un mayor detalle los puntos anteriores. En primer lugar, en las secciones II y III se profundizará en las funciones de la universidad para evidenciar la conveniencia de que las universidades públicas andaluzas (y españolas)

adopten políticas de defensa y fomento del software libre y en general del libre intercambio de la información y del conocimiento. En las secciones siguientes, se analizará con mayor profundidad el alcance de las dos medidas llevadas a cabo recientemente por la OSLUCA, que han sido comentadas más arriba.

## II. Sobre las Funciones de la Universidad

Los orígenes históricos de la Universidad se remotan a la última etapa de la Edad Media. En Europa, con la maduración del sistema feudal, el crecimiento sostenido de la economía y el incremento de la población, sobrevino el despertar de las ciudades y el de una nueva clase social, la burguesía. En este contexto, tuvo lugar la aparición de la figura del intelectual, persona se dedicaba a cultivar el conocimiento y a vivir del producto de su saber. Aunque la mayoría de ellos fueran jurídicamente clérigos y estuviesen al servicio de la Iglesia, el intelectual consideraba su oficio más como un fin en sí mismo que como un medio para llegar a Dios. Como el resto de los gremios urbanos, los intelectuales vieron la necesidad de impulsar su espíritu corporativo y de unirse para proteger sus intereses, lo cual dio origen a las universidades.

Fundadas desde el clero entorno a catedrales y escuelas episcopales, las universidades adquirieron pronto un cierto grado de autonomía que las situó a caballo entre el laicismo y la dependencia eclesiástica, pero su excesiva sumisión a al saber establecido y a la ortodoxia religiosa las convirtió en centros de enseñanza dogmática de saberes heredados de la antigüedad, ajenos a la revolución científica que se produjo en el siglo XVII y a su desarrollo en el siglo XVIII. Con la llegada de la Ilustración se fundamentó la idea de una escuela pública con procedimientos científicos y dirigida por expertos y durante los siglos XIX y XX, se extiendió la educación en todos los niveles sociales, decrementándose drásticamente el grado de analfabetismo y situándose, en este contexto, la universidad como entidad educativa superior.

En la actualidad, las Universidades Públicas españolas se conforman como entidades dotadas de personalidad jurídica y patrimonio propio que, de acuerdo con la Constitución, gozan de autonomía administrativa en el marco de lo dispuesto en la Ley Orgánica de Universidades. Y anuque su primera función sea el jugar el papel de centros de formación superior, éstas trascienden el mero papel educativo, adquierendo una dimensión más amplia que se articula en distintos ejes:

1. La formación superior, en particular la capacitación para el ejercicio de actividades profesionales.
2. La función de motor para la investigación, el desarrollo y la transmisión de la ciencia, la técnica y la cultura.
3. La aplicación de los conocimientos superiores de los que son depositaria, para la promoción del bien social, económico y cultural de la comunidad social en la que se sumergen.
4. El impulso de valores éticos como la libertad, la solidaridad, la cooperación, la justicia, el pluralismo, etc.

De estas cuatro funciones, que se reflejan, de una u otra forma, en los estatutos de todas las universidades públicas andaluzas, son las dos primeras (formación e investigación) las más conocidas, por lo que no entraremos aquí en mayores detalles. Pero sí puede ser oportuno profundizar aquí en las dos últimas funciones de las antes enumeradas, teniendo en cuenta su importante papel para dotar a las universidades públicas de su sentido social. Si tomamos como ejemplo a los estatutos de la Universidad de Cádiz[2], encontramos que, en su Artículo 2, define sus diez fines, entre los cuales se encuentran varios puntos relacionados con las funciones de desarrollo social del entorno que rodea a la universidad y de impulso de valores éticos positivos:

- Promover l*a aplicación práctica del conocimiento al desarrollo social*, cultural y económico y al bienestar de la Comunidad y de sus ciudadanos.
- Acoger, defender y promover los valores sociales e individuales que le son propios, tales como la *libertad, el pluralismo, el respeto de las ideas y el espíritu crítico*, así como la búsqueda de la verdad.
- Atender y prestar apoyo a todos aquellos aspectos relativos al *desarrollo científico, técnico y cultural de la Comunidad Autónoma de Andalucía*.
- Fomentar las *relaciones con otras universidades, Centros de educación superior y Centros de investigación*
- Promover la *conciencia solidaria* mediante una formación integral de la Comunidad Universitaria que consista en favorecer la información, la puesta en marcha de iniciativas, la promoción, la sensibilización y la actuación en problemas de justicia social, solidaridad y cooperación.

En cuanto a otras universidades andaluzas (y españolas), entre sus funciones, tal y como son definidas en sus estatutos, se recogen fines similares a los anteriores en cuanto al desarrollo del entorno local y de fomento de valores éticos.

## III. Sobre la relación entre Universidad y Software Libre

En la sección anterior se definió la universidad como un ente administrativo autónomo con múltiples fines (formativos, de investigación, sociales, éticos). Pasaremos ahora a estudiar la relación entre Universidad y Software Libre desde el punto de vista de cada uno de los aspectos anteriores.

### *III-A. La Universidad como Administración Pública*

En primer lugar las universidades tienen una serie de obligaciones heredadas de su carácter de administración pública, como pueden ser la de administrar su patrimonio racionalmente o la de facilitar a sus usuarios toda la información institucional de forma pública y transparente.

Si hablamos desde una perspectiva meramente economicista, debemos destacar una ventaja que suele asociarse comúnmente al uso de software libre: la reducción de costes que puede conllevar no ya su gratuidad, sino las consecuencias que se derivan de su libertad de uso, copia y adaptación y que nos garantizan que, incluso aunque el software no se adquiera gratuitamente, no serán necesarias grandes inversiones adicionales en estos capítulos. Cuando adquirimos software libre poseemos realmente el producto en el sentido de la libertad para utilizarlo, copiarlo o modificarlo de la forma en que creamos más adecuada, siendo esto último especialmente interesante en el ámbito universitario, que cuenta con personal sobradamente capacitado para adaptar el código a las características deseadas, sin estar atados a la voluntad o de una empresa externa a la universidad que no lo hará si no es en defensa de sus propios intereses comerciales y que, de hecho, impedirá que cualquier técnico de la universiad pueda modificar y mejorar el código, por muy capacitado que éste se encuentre.

El coste de uso del software propietario no se debe solamente al de adquisición de licencias, sino también a las restricciones de uso y copia que éstas conllevan. En efecto, al pagar una licencia de software propietario solamente se adquiere el permiso de ejecución bajo determinadas condiciones restrictivas, por ejemplo, la limitación a un número máximo de usuarios o prohibición a instalarlo en más de un ordenador si no se pagan nuevas licencias, aunque lo utilice una única persona. Un caso muy frecuente en la universidad es el de las licencias de Campus, que otorgan permiso para ejecución a todos los miembros de la institución pero solamente por un periodo de tiempo concreto, después del cual la entidad deberá volver a abonar el “alquiler”.

Otro consecuencia relacionada con su carácter de administración pública es que la universidad, como fuente de un importante caudal de información, debe ofrecer las máximas garantías de que toda ella estará disponible para sus miembros, tanto ahora como en el futuro, sin que sea una condición necesaria el que todos ellos dispongan de un mismo producto comercial. Para ello, la infomación no debe estar codificada en ningún formato cuyas especificaciones sean, total o parcialmente, secretas o puedan llegar a serlo. En lugar de ello, los documentos institucionales deberían ser codificados utilizando un formato abierto y neutral, de tal forma que todos los fabricantes puedan implementarlos en igualdad de condiciones y que cualquier persona tenga a su disposición, hoy y mañana, toda la información necesaria para recuperarlos. Este es, precisamente, el objeto de la “Normativa de Intercambio de Información Institucional”, recientemente aprobado por el Consejo de Gobierno de la Universiad de Cádiz y del que hablaremos con más detalle en el apartado IV-A.

### *III-B. La Universidad como Institución Educativa*

En todos los niveles educativos, en particular en el universitario, el uso en las aulas de las tecnologías de la información y de la comunicación aumenta día a día, abriendo las puertas a nuevas oportunidades y a terrenos inexplorados.

Para ello, el software libre constituye el marco idóneo, por numerosos motivos, como está haciendo patente la experiencia de la Junta de Andalucía en todos los niveles educativos previos al universitario. Mas, en la universidad pública andaluza, sigue imponiéndose un modelo docente basado en el uso de software privativo, bien sea dentro del aula de informática, bien como medio para intercambiar información o bien con algún otro fin. Esta elección se ampara en la libertad de cátedra del profesor y se gúia por consideraciones (subjetivas, en mayor o menor medida) tales como la calidad del producto, la adecuación a la asignatura o quizás la comodidad o la simple inercia.

Pero raramente se tiene en cuenta una importante cuestión: la velocidad con la que evoluciona la sociedad de la información puede hacer que tecnologías que hoy son hitos incuestionables sufran mañana severos cambios e incluso lleguen a ser superadas y olvidadas en cuestión de años. Por tanto una formación basada en la excesiva dependencia de una única herramienta comercial, puede llegar, con el tiempo, a ser ineficaz. Los estudiantes deben estar formados en habilidades generales, en conocimiento neutral, y no en los productos concretos de una marca comercial. Sólo de esta manera se garantizará el carácter universal de los conocimientos adquiridos y se evitará que la no disponibilidad de un producto o sus carencias evidencien las lagunas del proceso formativo.

El software libre, como contrapunto, permite disponer de varias, frecuentemente muchas herramientas a la vez, con frecuencia complementarias o capaces de operar entre sí, cada una de las cuales contará con sus puntos fuertes y sus debilidades. Aunque el profesor se decante por una concreta, siempre podrá ofrecer a sus alumnos la enriquecedora posibilidad de experimentar con otras, de resolver un mismo problema desde distintas perspectivas y de saciar su curiosidad a aquellos que cuenten con mayores inquietudes.

Por otro lado, es importante destacar que la dependencia de una herramienta comercial en el ámbito educativo conlleva problemas éticos añadidos, puesto que de forma irremisible provoca en el alumnado la seducción por una marca cuyo precio hará que, en la mayoría de los casos, no pueda ser adquirida legalmente. De esta forma, se incita a los alumnos a una actividad perseguida por la ley como es la copia ilegal de programas de licencia privativa. En el otro extremo se sitúa el caso del software libre permitiendo que el profesor comparta con sus alumnos, con toda legalidad, las herramientas utilizadas (quizás, acompañadas de material docente propio), permitiéndoles, por ejemplo, reproducir en sus hogares el entorno de trabjo del aula.

### *III-C. El Software Libre y la Investigación*

Con la revolución científica en la Europa de los siglos XVI y XVII, tuvo lugar el nacimiento de la investigación, tal y como hoy la conocemos. Entre los pilares de esta revolución científica, que por cierto no se desarrolló dentro de las universidades, sino más bien bajo la protección del mecenazgo y de academias extra universitarias, se encuentran el escepticismo crítico (ninguna teoría es aceptada si no es demostrada científicamente, por medio de la razón) y la necesidad de publicación (revisión de las teorías por parte la comunidad científica, con el fin de filtrarlas y publicarlas para aumentar el acervo del conocimiento). El propio Isaac Newton reflejó la importancia de la publicación del conocimiento cuando expresó “si he llegado a ver más lejos que otros, es porque me subí a hombros de gigantes”. Si Newton no hubiera podido acceder libremente a los trabajos de Kepler y de Galileo, probáblemente nunca hubiera llegado a formular su teoría de la gravitación y sin ella, el hombre nunca hubiera llegado a la Luna.

Como investigadores, las cualidades anteriores (escepticismo crítico y necesidad de publicación) resultan naturales para los profesores universitarios, que basan, de hecho, en la cantidad de trabajos publicados y en la calidad de los mismos sus posibilidades para una posible mejora en sus condiciones laborales. Es, por lo tanto, evidente para el profesorado la idea de que la publicación del conocimiento, la reutilización y mejora de la información, son aspectos muy positivos, en los que se cifra el avance del saber. Así, debería parecerles natural que el mismo camino se aplicara en todos los ámbitos del conocimiento y les debería resultar chocante que haya entidades que se niegen a hacer pública, por ejemplo, la información sobre cómo funciona un dispositivo o el algoritmo que se esconde detrás de un programa informático. Mientras están apoyándose, para su propio lucro, en los hombros de gigantes del saber científico acumulado durante siglos sin aportar, a cambio, conocimiento a la humanidad.

Pero, resulta aún más preocupante que Andalucía, España y Europa destinen ingentes cantidades económicas para la adquisición de software privativo que, además de no aportarles conocimiento, son transferidas a multinacionales foráneas que nos venden productos para los cuales existen alternativas casi gratuitas que podemos adaptar y mejorar por nosotros mismos. Y esto es todavía más grave si pensamos en que estos enormes fondos transferidos fuera de nuestras fronteras podrían ser invertidos en I+D, para acercar los niveles de inversión al de otros paises, como Estados Unidos y reducir el margen tecnológico que nos separa.

### *III-D. La Universidad como Impulsora de su Entorno Social*

La sociedad actual se encuentra en un proceso de cambios que muchas personas no dudan en calificar como una tercera revolución industrial. Si la primera revolución industrial estuvo sustentada en la máquina de vapor y la segunda lo estuvo en el petróleo y en la electricidad, la información sería la materia prima en la que se fundamenta esta nueva revolución. Nuestra capacidad para, generar información, gestionarla y difundirla utilizando las nuevas tecnologías (las que se conocen como tecnologías TIC) es inmensa, hasta tal punto que el conocimiento se convierte en un recurso externo al ser humano, inabarcable. En tal situación, conocida como *sociedad de la información*, la posesión del conocimiento se convierte en un factor que, de cara al incremento de la productividad de la economía, puede ser tanto o más importante que elementos clásicos como el capital, trabjadores, etc.

En estos momentos, la sociedad andaluza está apostando fuerte por no perder el tren de la alfabetización tecnológica, cuya meta es el acceso de los ciudadanos a la sociedad de la información. Pero la alfabetización tecnológica no tiene sentido si no es construida como un proyecto colectivo, basado en el conocimiento universal

y no en la adquisición conocimientos particulares, basado en la libertad y en la igualdad de los ciudadanos y no en la dependencia tecnológica de una o varias empresas comerciales. Un proyecto de esta envergadura no pasa (no debería pasar) por el sometimiento a multinacionales para las que sus intereses personales priman muy por delante del de los ciudadanos.

Las Universidades Andaluzas en ejercicio de las funciones sociales que les son propias (y que fueron analizadas en la sección II), están obligadas por sus estatutos a realizar un esfuerzo encaminado a aplicar el conocimiento superior del que son depositarias para el bien de los ciudadanos. Y esto significa apoyar el proceso de alfabetización tecnológica de la sociedad, "atender y prestar apoyo a todos aquellos aspectos relativos al desarrollo científico, técnico y cultural de la Comunidad Autónoma de Andalucía"[1].

Por otra parte, la extensión del software libre implicaría el desarrollo tecnológico local, aprovechando un nuevo modelo en el negocio de las tecnologías de la información, en el que las empresas tecnológicas regionales puedan actuar en auténtica competencia y sin la subordinación tecnológica a empresas multinacionales. Este nuevo modelo, explota el hecho de que una parte fundamental del negocio del software está está en los servicios de soporte y personalización, según las necesidades de los usuarios, más que en la venta de software en sí mismo. Y las empresas locales, contando con la cercanía y la confianza como puntos fuertes, se encuentran mucho más capacitadas para explotar este modelo.

### *III-E. Aspectos Éticos del Software Libre*

Si atendemos a los aspectos éticos, encontraremos de nuevo que el software libre se encuentra en sintonía con lo proclamado por los estatutos de las Universidades Andaluzas y Españolas. En efecto, apoyar el software y el conocimiento libres significa fomentar aspectos éticos positivos, como la colaboración (en la creación o reutilización del conocimiento, en el desarrollo o mejora de software, etc), la solidaridad (disponibilidad y libertad de copia de la información para cualquier persona que la pudiera necesitar), el conocimiento sin trabas, etc.

El software propietario, por contra, produce un importante enfrentamiento ético entre lo que dice la ley (es ilegal la copia de información protegida por copyright si su autor no lo autoriza expresamente) y lo que puede sugerir en ocasiones la conciencia humana (no es malo compartir información con un hermano). Hay incluso quien aprecia una doble moral, por parte de empresas comerciales que no autorizan la copia ilegal de sus programas a la vez que, sin que lo reconozcan expresamente, se ven beneficiadas por su difusión, aun de forma ilegal.

[1]El texto entrecomillado está extraído literalmente de los Estatutos de la Universidad de Cádiz[2]

## IV. Avances Recientes en la Universidad de Cádiz

En los últimos meses, los pasos de la Oficina de Software Libre de la Universidad de Cádiz se han encaminado hacia las dos acciones concretas que detallamos en esta sección

### *IV-A. Normativa de Intercambio de Información Institucional*

Como fruto de su política de apoyo al software libre, el 27 de septiembre de 2004, el Consejo de Gobierno de la Universidad de Cádiz aprobó, a instancias de la OSLUCA, la "Normativa de intercambio de información institucional en la Universidad de Cádiz"[3]. La normativa entró en vigor a partir de su publicación el pasado 11 de Octubre de 2004 en el Boletín Oficial de la Universidad de Cádiz.

Este trascendental documento, sin parangón entre las administraciones públicas españolas, implica en que, después de un periodo de transición, todos los documentos oficiales que sean colocados en internet, enviados por correo electrónico o compartidos por cualquier otro medio deberán haber sido codificados usando un formato abierto. Esto implicará una auténtica revolución en el seno de la Universidad de Cádiz y hará necesario realizar un importante esfuerzo para concienciar a todos sus miembros (profesores, estudiantes, personal de administración y servicios) y capacitarlos para producir documentos en formatos abiertos.

La normativa refleja explícitamente el razonamiento que ha sido expuesto en la sección III-A sobre la necesidad de que la universidad, como administración pública, ofrezca a sus integrantes garantías de que la información estará siempre disponible para todos sus integrantes y concluye:

"Si no se utilizan estándares públicos para el intercambio de datos oficiales, la elección por parte de la UCA de un determinado formato conllevará la dependencia de determinado proveedor de software, limitando la libertad de elección de sus integrantes, atrapándolos a un único proveedor y obligándolos a adquirir sus productos y depender de ellos. Resulta impensable que los documentos que sean emitidos institucionalmente por una administración pública se encuentren codificados en un formato propietario de un único proveedor u organismo externo sin cuya licencia sea imposible el acceso a ellos."

Y como consecuencia, dispone: “Cualquier documento emitido por un organismo de la UCA que esté dirigido oficialmente a miembros de ella deberá estar codificado en un formato abierto, siempre que exista alguno adecuado para el tipo de documento del que se trate, o al menos deberá acompañarse de una versión en formato abierto, independientemente del medio empleado para su emisión: correo electrónico, página web, TAVIRA, etc.”.

Por último, señala que: “Será tarea del CITI el mantener una lista actualizada de formatos abiertos para diferentes tipos de documentos, a los cuales deberán atenerse las publicaciones de documentos institucionales.”.

Por supuesto, dentro del CITI, la tarea de especificar cuáles son los formatos adecuados para el intercambio de la información descansa en la OSLUCA. A esta tarea nos hemos dedicado en las últimas semanas, propiciando el debate sobre este asunto tanto en los foros de nuestra página web como en nuestra lista de correo. La decisión, que hasta el momento hemos restringido al caso de formatos de documentos de texto, ha distado mucho de ser sencilla, puesto que debemos ser muy cuidadosos para asegurarnos de que los formatos abiertos que especifiquemos están lo suficientemente extendidos y en un grado suficiente de madurez como para poder imponer su uso a todos los miembros de nuestra universidad.

Se trataba, en definitiva, de elegir formatos abiertos para los que existieran programas capaces de leerlos y de generarlos y tales que estos programas cumplieran los siguientes requisitos:

1. Ser libres
2. De buena calidad
3. Estar muy difundidos
4. Disponibles en las plataformas más usadas en puestos de trabajo.

Solamente se encontraron dos formatos que reunieran todos estos requisitos: el texto simple y HTML/XHTML (en cualquiera de las versiones publicadas por el W3C). El primero de ellos, aunque adecuado para mensajes sencillos, como el correo electrónico, no es válido para documentos más elaborados, que requieran el resaltado de texto. El segundo, aun permitiendo la creación de documentos más elaborados, tiene como inconveniente el estar compuestos, frecuentemente, por distintos ficheros (gráficos, hojas de estilo, etc) lo que puede provocar confusión en el usurario común. Esta confusión puede verse incrementada por herramientas de generación de HTML que están muy extendidas y generan código no compatible con la especificación estándar.

Existen otros tipos de ficheros que cumple a la perfección los cuatro requisitos anteriores, salvo el tercero (estar muy difundido). El más importante de ellos es, sin duda, SXW, el formato nativo de OpenOffice.org para documentos de texto, aunque existen otros formatos que entrarían en este saco, como DOCBOOK, LaTeX, etc.

Basado en XML, ha SXW ha sido escogido por Oasis como base para la creación de un estándar para formatos utilizados en las suites ofimáticas. Y aunque sea un formato poco extendido entre los usuarios del sistema operativo dominante, la libertad y la calidad de OpenOffice.org unida a una política de formación adecuada entre los miembros de la UCA podría convertirlo, a medio plazo, en el formato idóneo para documentos estructurados que sean modificables por los usuarios.

Pero mientras llega este momento, es imprescindible facilitar la transición al usuario medio habituado a usar herramientas propietarias para el intercambio de documentos. Una alternativa muy interesante sería el formato PDF, si no fuera por que, en sentido estricto, no podemos considerarlo como un formato libre, entre otros motivos por ser un formato dependiente del arbitrio de una único fabricante y por acarrear diversas patentes (aunque sean patentes de tipo “royalty-free”). Aun así, este formato es muy adecuado para la creación de documentos cuyos destintarios no deban modificarlos. Teniendo en cuenta que esta característica no está disponible en ningún formato abierto que esté los suficientemente extendido, junto a que es un formato cuyas especificaciones son públicas y maduras y puesto que existen programas libres, de buena calidad y disponibles en diversas plataformas capaces de generarlos y leerlos, por la conjunción de estos motivos, se ha creido conveniente el relajar temporalmente la normativa de intercambio de información institucional para aceptar documentos de tipo PDF, mientras no existan estándares abiertos equivalentes que estén lo suficientemente extendidos.

Por tanto, la OSLUCA lanzará en breve una campaña informativa para la puesta en marcha de la normativa de intercambio de información institucional que fue aprobada por el Consejo de Gobierno de la UCA, en la que se recomendará el uso de ficheros SXW para documentos modificables y PDF para documentos que no deban ser modificados, se permitirá el uso de ficheros de texto simple y HTML y se prohibirá el uso de formatos cerrados, como ficheros .DOC, o formatos públicos que dependan de un fabricante comercial, como RTF, cuya madurez no pueda ser constatada o para los que existan alterntivas razonables.

### *IV-B. El sistema operativo* Guadalinex_UCA *GNU/Linux*

El últimos años, ha empezado a ser frecuente la aparición de distribuciones de GNU/Linux promovidas por las administraciones públicas, las más conocidas

de las cuales son gnuLinex, producida por la Junta de Extremadura, y Guadalinex, creada por la Junta de Andalucía. De forma parlela, distintas universidades han optado por la distribución entre sus integrantes de sistemas operativos derivados de GNU/Linux, como SILU, por parte de la Universidad de Las Palmas de Gran Canarias, o gnUAMix, por la Universidad Autónoma de Madrid.

Esta multiplicación de distribuciones de GNU/Linux no ha estado exenta de críticas, que pueden haber estado fundamentadas en mayor o menor medida. Entre ellas, se les ha achacado el provocar una imagen de dispersión y falta de unidad ante los ciudadanos y el derrochar esfuerzos inútiles en vez de aportaciones a la comunidad, buscando solamente el beneficio político de incrustar el logotipo de la administración pública en el sistema operativo.

Aun consciente de todos estos inconvenientes e intentando hacer un esfuerzo por soslayarlos, la OSLUCA ha encontrado suficientes razones que justifican la creación de un sistema operativo "propio".

En primer lugar, se ha evaluado positivamente el impacto beneficioso que esto tendría entre los integrantes de nuestra universidad y en su entorno. Por una parte, la creación de Guadalinex_UCA evidencia que la fabricación de tecnología de alta calidad basada en software libre se encuentra al alcance de cualquier entidad (lo que se puede relacionar con los aspectos sobre la investigación comentados en la sección III-C). A su vez, la personalización de la tecnología tendría la fuerza subjetiva de hacerla ver como propia a personas que puedan estar muy distantes de la filosofía de la libertad del conocimiento. Relacionado con lo anterior se encuentra el que la posibilidad que tienen las distribuciones como Guadalinex_UCA de arrancar directamente en modo de evaluación del CD para que cualquier persona pueda evaluar las herramientas en él incluidas, es una útil herramienta para difundirlas y para acercarlas personas que, de otro modo sentirían un mayor respeto llevar software desconocido a su ordenador.

Por supuesto, una de las razones fundamentales para haber creado una adaptación del Guadalinex está relacionada con la posibilidad de incluir programas no incluidos por defecto en la distribución original. Así, se han eliminado de la distribución original aqullos programas que no se estimaron prioritarias en el ámbito universitario y se ha añadido una selección de herramientas científicas y técnicas que se han considerado de utilidad en nuestra Universidad, agrupándolas dentro de una entrada propia en el menú de aplicaciones.

El que profesores y alumnos puedan comprobar de primera mano que existe una amplia gama de aplicaciones libres aplicables en sus áreas de conocimiento enlaza con lo que comentábamos en el apartado III-B y es un paso en la dirección del uso de software libre en la educación. Y en este mismo sentido, los profesores que deciden usar una herramienta libre en el aula de ordenador podrán entregar, con toda la legalidad, a sus alumnos una copia de Guadalinex_UCA para que puedan seguir utilizándola fuera del ámbito del aula sin que restricciones económicas o legales puedan impedirlo. Se han incluido en Guadalinex_UCA algunas aplicaciones libres que están empezando a ser usadas este curso en las aulas, como consecuencia de las iniciativas que empiezan a nacer en algunos departamentos de la UCA.

Por otra parte, como es patente a partir del nombre que hemos dado al sistema operativo, se ha intentado evitar en lo posible la imagen de fragmentación del software libre, teniendo gran cuidado en dejar muy claro que, con la publicación de Guadalinex_UCA, la Universidad de Cádiz no pretende crear una nueva distribución de GNU/Linux independiente de las demás, sino adaptar y personalizar un sistema operativo ya existente, Guadalinex, que a su vez no es más que una localización de Debian GNU/Linux, una de las distribuciones más extendidas en el mundo y aquella que cuenta con mayor cantidad de desarrolladores voluntarios. Al enfatizar la relación entre Guadalinex_UCAy Guadalinex estamos, además, respaldando el camino emprendido en la comunidad autónoma de Andalucía hacia la alfabetización tecnológica, en el sentido que comentábamos en el apartado III-D.

Por último, es conveniente hacer patente que la publicación de Guadalinex_UCA no es un punto final, sino un proceso abierto en el que muchos aspectos son mejorables. Y no sólo desde el punto de vista de las correcciones necesarias en aspectos técnicos, sino porque ahora se hace presente el reto el involucrar a la comunidad universitaria para que entiedan a Guadalinex_UCA como algo propio, que los profesores sepan que es una herramienta a su disposición para difundir los programas que usen en el aula y el material docente que los acompañe, que los estudiantes puedan ver incluido el fruto de su trabajo y el de sus compañeros y que sea, realmente, un motor para acercar el software libre a la comunidad universitaria.

## REFERENCIAS

[1] Consejo de Gobierno de la Universidad de Cádiz, "Declaración de apoyo al software libre en la uca," http://softwarelibre.uca.es/documentos/creacion_osluca, Marzo 2004.

[2] Universidad de Cádiz, "Estatutos," http://www.uca.es/normativa/paginas/estatutos_UCA.html, Julio 2003.
[3] Consejo de Gobierno de la Universidad de Cádiz, "Normativa de intercambio de información institucional en la universidad de cádiz," .

# Aproximación al nivel de seguridad de soluciones basadas en Software Libre

Alvaro López Ortega, <alvaro@gnu.org>

***Resumen*— Se trata de un estudio de las características del Software Libre y las consecuencias directas e indirectas que este movimiento de colaboración global tiene sobre la calidad y seguridad del software que en él se genera. Se presentarán datos sobre productos de gran difusión en los últimos años y sus incidencias de seguridad, con el fin de estudiar la posible relación de estos problemas con el modelo de desarrollo en que se basa cada uno de ellos. El objetivo final es mostrar si este modelo de desarrollo de software es más o menos seguro que el clásico modelo de Software Cerrado.**

***Palabras clave*— Seguridad, Software Libre, vulnerabilidad**

## I. Introducción

En este artículo se va a realizar una aproximación a la valoración de los niveles de seguridad que se alcanzan en los productos de Software Libre. El medir el nivel de seguridad de un paquete de software no es una tarea sencilla, en él intervienen infinidad de parámetros con una capacidad de variación muy grande sobre la evaluación final. La metodología con la que se intentará evaluar la calidad de software producido en un modelo de desarrollo de Software Libre se basará en su comparación con paquetes de software cerrado/propietario/privativo (desde ahora nos referiremos a este modelo de desarrollo de software como Software Cerrado). De esta forma, este trabajo lejos de intentar cuantificar la calidad de cada uno de los modelos de software con una valoración final para cada uno de ellos, los comparará entre sí en busca de encontrar el mejor de los dos. Por supuesto, el resultado y la valoración final de este, habrá que acotarla a los ámbitos en los que se ha centrado el estudio, es decir, ni se han expuesto todos los modelos de desarrollo de software, ni se han comparado todos los paquetes de software de cada uno de los dos modelos estudiados. También hay que tener en cuenta que en los últimos años, está creciendo el número de programas de Software Cerrado que incorporan en su interior fragmentos de código y bibliotecas libres - normalmente licenciadas bajo BSD o LGPL. Esto hace un poco más difícil aún el poder comparar los dos métodos de desarrollo.

El Software Libre es un movimiento de colaboración global en el que se desarrollan programas de una forma abierta. El código fuente de estos programas está disponible para que cualquier persona lo descarge, lea, use, modifique o redistribuya. El desarrollo se basa en la comunicación por medio de Internet, y se realiza entre grupos de gente que en muchas ocasiones no se conocen en persona. Al contrario que en el desarrollo de Software Cerrado, en el Software Libre no se sigue ninguna especificación de desarrollo basada en la Ingeniería del Software: ¿El no seguir un conjunto de normas estrictas hará que se genere un software más inseguro?.

Por otro lado, en el desarrollo de Software Libre se van liberando versiones de un programa durante su desarrollo. En muchos casos estas versiones “beta” - no terminadas - de programas se utilizan. En el caso del modelo de desarrollo del Software Cerrado, es conocido que con frecuencia existen problemas de falta de tiempo en el desarrollo: una práctica común al contratar un servicio de esta clase es firmar una fecha máxima de finalización del proyecto; en otras ocasiones, son los aspectos relacionados con el marketing y promoción del producto los que requieren que se termine de desarrollar antes de una fecha marcada. Desde este punto de vista, es posible que haya Software Cerrado en estado “beta” a causa de estas presiones temporales. Dada esta circunstancia, ¿Hay alguna diferencia entre la seguridad de un programa generado por el modelo de Software Cerrado, con una versión no terminada de Software Libre?.

Ahora bien, todos estos pequeños factores no son demasiado relevantes en comparación con, por ejemplo, la motivación que lleva a desarrollar el software. En el caso del modelo de Software Cerrado, la motivación es puramente económica: se desarrolla un paquete de software a cambio de una compensación económica. Un desarrollador de Software Libre puede realizar esa tarea por distintas motivaciones, mayoritariamente éticas o económicas. Si es por razones éticas, comparte el punto de vista más científico de que el software es conocimiento, y como tal conocimiento no debería ocultarse, sino difundirse para progresar. Si por el contrario lo hace por razones económicas, es porque cree en que es un nuevo modelo de desarrollo de software sostenible y rentable, con el que puede obtener un mejor resultado que con el modelo Cerrado, y en algún caso situarse en un mercado en el que de otra forma sería mucho más difícil acceder. En cierta forma, al publicarse el trabajo de un desarrollador - grupo, o empresa - de Software Libre se está evaluando en público su calidad. Al dejar el código fuente accesible, permite que cualquier persona pueda evaluar su calidad, y por lo tanto, su calidad como empresa o profesional.

Además de todos estos factores, existen otros de igual importancia: socialmente hay una gran permisibilidad con la copia ilegal de software, la adopción de soluciones basadas en Software Libre son relativamente recientes en algunos sectores, etc. Según los datos de BSA - Business Software Alliance

ISBN 84-88783-71-X©2004

- existen paises en los que se copia ilegalmente hasta el 94 % del software que se utiliza. Esta cultura de copiar ilegalmente

TABLA I
BSA - DATOS SOBRE COPIA ILEGAL

| Pais | 2001 | 2002 |
|---|---|---|
| Vietnam | 97 % | 94 % |
| China | 94 % | 92 % |
| Indonesia | 89 % | 88 % |
| .. | .. | .. |
| Grecia | 66 % | 64 % |
| España | 51 % | 49 % |
| Portugal | 42 % | 43 % |
| Francia | 40 % | 46 % |

Software Cerrado también puede estar relaccionada con el nivel de seguridad que de él se desprende: Hoy en día, existe un grupo indeterminado de gente y empresas que utilizan Software Cerrado ilegalmente; gracias a este hecho, la base instalada de estos productos - y por tanto, de conocimiento a su alrededor - es mayor de lo que sería sin la existencia de la copia ilegal de Software Cerrado. ¿Tiene esto alguna relación con la calidad del software?. Una mayor difusión y por tanto, una mayor base de conocimientos sobre los productos cerrados hace suponer en un primer momento que deberían ser más seguros.

## II. SISTEMAS OPERATIVOS

Un buen punto de partida en la comparación de la seguridad de los programas producidos en el modelo Libre y Cerrado pueden ser los sistemas operativos. En los últimos años se han revelado múltiples vulnerabilidades tanto por el lado del Software Libre en productos como: GNU/Linux o FreeBSD, como en el del Software Cerrado con productos como: Microsoft Windows o Cisco IOS. La ventaja de esta comparación en especial es que se trata de un elemento imprescindible para cualquier dispositivo; gracias a esto, los datos tomados serán mucho más representativos que en otras posibles comparaciones.
En la actualidad existen organizaciones dedicadas a la seguridad informática en diferentes ámbitos y a la publicación de vulnerabilidades descubiertas en paquetes software. En concreto, a continuación se presentarán las estadísticas obtenidas de los datos de una de ellas, BugTraq [1]. Durante años, esta organización ha publicado los problemas de seguridad de todo tipo de software. La tabla que se expodrá a continuación muestra el número de vulnerabilidades de los sistemas operativos más extendidos hoy en día.

TABLA II
ESTADÍSTICAS SOBRE VULNERABILIDADES

| Sistema Operativo | 97 | 98 | 99 | 00 | Total |
|---|---|---|---|---|---|
| Windows NT | 4 | 6 | 99 | 34 | 143 |
| Windows 9x | 1 | 1 | 46 | 11 | 59 |
| Solaris | 24 | 31 | 34 | 6 | 95 |
| IRIX | 26 | 13 | 8 | 3 | 50 |
| HP-UX | 8 | 5 | 7 | 3 | 23 |
| MacOS X | 0 | 1 | 6 | 0 | 7 |
| — | - | - | - | - | – |
| GNU/Linux (*) | 10 | 23 | 84 | 30 | 147 |
| Debian | 2 | 2 | 29 | 5 | 38 |
| Red Hat | 5 | 10 | 38 | 17 | 70 |
| FreeBSD | 4 | 2 | 18 | 6 | 30 |
| OpenBSD | 1 | 2 | 4 | 2 | 9 |

Conviene aclarar cómo hay que interpretar los datos anteriores. La primera parte de la tabla contiene los sistemas operativos de Software Cerrado, y la segunda parte los de Software Libre. En esta última parte perace una fila con el título “GNU/Linux (*)”, eso es porque en esta entrada se contabilizan todos los problemas de seguridad de GNU/Linux; se trata de la suma de todas las vulnerabilidades diferentes aparecidas en todas las distribuciones, pero sin duplicar su conteo si aparece dos veces el mismo problema en dos distribuciones diferentes.

Por otro lado hay que estudiar las diferencias entre cada uno de los sistemas operativos:

- Los sistemas operativos cerrados tipo UNIX en muchas ocasiones ejecutan algunos programas libres. Típicamente sendmail o bind.
- El tamaño de un sistema operativo Windows y uno GNU/Linux es grandísimo. Por ejemplo, la distribución de GNU/Linux “Debian” actualmente incluye más de 8700 paquetes.
- Los Unix libres - GNU/Linux, y *BSD - comparten gran parte de sus aplicaciones.

El tamaño de los sistemas operativos es un factor crítico en esta comparación. En estas estadísticas, se ha contabilizado las vulnerabilidades de los sistemas operativos según se distribuyen. Esto quiere decir que se ha evaluado a los sistema libres con varios miles de aplicaciones y a los cerrados únicamente con algunas decenas. El modelo económico del Software Cerrado se basa en la venta de aplicaciones que se ejecutarán sobre un sistema operativo, pero no en la distribución conjunta de estas con el propio sistema. Existen infinidad de aplicaciones que resultan básicas para el uso cotidiano de un ordenador, pero que en muchos sistemas operativos no se distribuyen por defecto: Herramientas de ofimática, lectores de correo electrónico, herramientas de trabajo con gráficos, etc. Todas las vulnerabilidades de este software no han sido contabilizadas en las estadísticas expuestas anteriormente. Por ejemplo, las incidencias de seguridad en Microsoft Exchange no fueron contadas ya que se entiende que este software no es parte

del sistema operativo. Por el contrario, las incidencias de sendmail si fueron contabilizadas ya que este servidor si que se distribuye conjuntamente con el resto del sistema operativo libre.

Además de la variedad de aplicaciones incluidas en una y otra clase de sistema operativo, hay que tener en cuenta que en la mayoría de los casos los sistemas operativos libres no incluyen una única opción para cada una de las herramientas. Por ejemplo, si se trata de un gestor de bases de datos normalmente incluirá MySQL y PostgreSQL, como entornos de escritorio GNOME o KDE, como navegador web Mozilla o Konqueror, como cliente de correo Kmail, Mutt o Evolution, etc.

También hay que tener en cuenta otros muchos factores: algunas de las vulnerabilidades son mucho más importantes que otras, no todas son explotadas por los atacantes de la misma forma, la velocidad en el que el desarrollador/fabricante del producto corrige y publica una solución, la velocidad con la que los administradores de sistemas o usuarios lo aplican, etc. Por esta razón, hay que tener en cuenta la forma en la que se trabaja con los informes de vulnerabilidades en cada uno de los casos. En los programas libres, al estar todo su código fuente disponible para realizar una auditoría es muy fácil estudiarlo en busca de algún problema de seguridad. Si se descubre un problema, este se hace público - por lo tanto, se contabiliza como tal - y a continuación se corrige y se libera una nueva versión del software. En el caso de software cerrado, si se realiza una auditoría de seguridad sobre su código, los posibles fallos y vulnerabilidades que se detecten serán corregidas, pero en muchos casos no se harán públicas - y por tanto, no serán contabilizadas a no ser que alguien detecte el error por medio de la experimentación o el uso de dicho software. Esta forma de auditar el software claramente favorece un menor número de vulnerabilidades públicas en el software cerrado.

La forma de distribuir las correcciones del software también suele ser muy diferente entre los sistemas operativos abiertos y cerrados. En el caso de los abiertos se tiende a publicar parches enfocados a la corrección de un único error. Estos parches se publican, y a continuación, se libera una nueva versión del programa, y posiblemente una nueva versión de los binarios de los paquetes para sus distribución. En el caso del software cerrado, las compañías tienden a recolectar un conjunto de parches y distribuirlos conjuntamente - algunos fabricantes, como Microsoft, los llaman Service Pack. Históricamente se ha podido comprobar que hay veces que estos paquetes de parches chocan con parches individuales o que incluso han incorporado nuevos problemas al arreglar los anteriores. Se ha generalizado esta situación, de forma que llegado al momento en el que la instalación de un navegador web en un sistema operativo cerrado haya cambiado el kernel del sistema sin previo aviso.

En la undécima conferencia anual de RSA se expusieron los datos sobre las intrusiones de seguridad durante ese año. En "Security in 2002 worse than 2001, exec says" [2] se exponen los siguientes datos sobre el número de ataques que recibieron cada uno de los sistemas operativos.

TABLA III
RSA: DATOS SOBRE INTRUSIONES

| Sistema Operativo | Nº ataques |
|---|---|
| Windows | 31 millones |
| Unix | 22 millones |
| Cisco IOS | 7 millones |

En estos datos todos los sistemas operativos Unix aparecen reflejados en el mismo valor. Entre ellos se encuentran tanto los sistemas libre: GNU/Linux y *BSD, como los de cerrados: Solaris, HP-UX, AIX, etc. Este hecho invalida en parte estos datos para los propósitos de comparar la seguridad del Software Libre y Cerrado ya que no es posible distinguir entre los ataques que sufrieron los sistemas libres. En el peor de los casos, y contabilizando todos los Unix como sistemas libres - lo cual no es cierto - los sistemas operativos cerrados hubiesen recibido 38 millones de ataques frente a los 22 millones de los sistemas libres.

Por otro lado existen datos que pueden ayudar a clarificar los anteriores. Netcraft, una empresa dedicada a la estadística y la seguridad en Internet ha publicado los datos sobre los sistemas operativos de los servidores de Internet en ese mismo periodo de tiempo [3].

TABLA IV
NETCRAFT: SISTEMAS OPERATIVOS DE SERVIDORES DE INTERNET

| Sistema Operativo | nº servidores | % |
|---|---|---|
| GNU/Linux | 6,116,811 | 35.73 % |
| Windows | 3,644,187 | 21.32 % |
| Otros | 3,802,268 | 21.24 % |
| Solaris | 3,484,135 | 20.35 % |
| Desconocido | 233,676 | 1.36 % |

Con estos datos, y suponiendo un total de 9 millones de servidores ejecutando un sistema operativo libre, la media sería de 2,4 intrusiones por cada sistema. En el caso de los sistemas operativos propietarios la media de intrusiones se supera el doble: 5,3. Este no sería un dato significativo si los sistemas operativos libres fuesen minoritarios y no gozasen de una amplia base instalada, pero basándonos en las estadísticas de NetCraft se puede ver como no es así y, al menos, hay igual número de unos y otros.

## III. EL TIEMPO: ES FUNDAMENTAL

Otro punto de interés respecto a la seguridad del Software Libre y el Cerrado es el tiempo que transcurre entre que una vulnerabilidad es descubierta, hasta que se publica una solución para el producto afectado. Es decir, la cantidad de tiempo que el sistema es abiertamente vulnerable - hasta ese momento lo había sido de igual forma, pero el problema no era público.

Existen estudios sobre la velocidad en que diferentes compañías arreglan y publican soluciones para sus productos. Por

ejemplo, SecurityPortal realizó un estudio sobre Microsoft, Sun y RedHat. [4]

TABLA V
SECURITYPORTAL: VELOCIDAD DE REACCIÓN

| Compañia | Intervalo vulnerable | Vulnerabilidades | Media |
|---|---|---|---|
| Red Hat | 348 | 31 | 11,23 |
| Microsoft | 982 | 61 | 16,10 |
| Sun | 716 | 8 | 89,50 |

En dicho estudio se puede ver como la compañía que tuvo un menor intervalo de vulnerabilidad fué Red Hat; esto quiere decir, a igualdad de atención en la actualización de un sistema Windows y uno GNU/Linux de Red Hat, el sistema no libre fué apróximadamente tres veces más vulnerable que el libre. Un comentario especial merece el caso del Solaris, el sistema operativo de Sun, ya que fué el que sufrió un menor número de vulnerabilidades, pero fué el que más tardó en arreglarlas, haciendo que su resultado final fuese muy inferior a la de las otras compañías. Por último hay que tener en cuenta que, en el caso de Red Hat, pueden transcurrir días desde que una vulnerabilidad de un programa libre es corregida hasta el momento en que las empresas dedicadas a las distribuciones adoptan la solución, reconstruyen sus paquetes y los ponen disponibles para sus usuarios; es decir, normalmente es posible disponer de una versión corregida del software antes de que la propia empresa publique su versión. Esto reduciría aun más el tiempo en el que un programa libre se encuentra comprometido por una vulnerabilidad.

Los datos anteriormente expuestos pueden tener más o menos relevancia en función de cuán importante es que un sistema se encuentre vulnerable. Si existe un bajo riesgo de que la vulnerabilidad sea atacada y/o las consecuencias del ataque son pequeñas, entonces no tendrán gran importancia. Si por el contrario, las consecuencias son más severas y/o existe un gran riesgo de que el sistema sea atacado, el tiempo en el que el sistema es vulnerable se vuelve fundamental para su seguridad. A este respecto, el CERT ha publicado un estudio[5]: consiste en la instalación de un sistema vulnerable y la posterior toma de datos para, de esta forma, ver cuanto tiempo tardaba en ser comprometido y de qué ataques era objeto. Los datos fueron los siguientes:

- A las 8 horas, se detectaron pruebas de vulnerabilidades en el RPC
- A los 21 días, se habían detectado 20 intentos de exploits conocidos
- A los 40 días, el sistema estaba comprometido por medio de una vulnerabilidad en el servidor POP3. El intruso instaló un sniffer, varias puertas traseras y modificó los logs del sistema.

Con estos datos, ya es posible valorar adecuadamente los del estudio de Security Portal, y comprobar como el sistema basado en Software Libre es efectivamente mucho más seguro que sus dos competidores del estudio basados en Software Cerrado.

## IV. VIRUS, GUSANOS Y TROYANOS

Los virus, gusanos o troyanos son también un aspecto a tener en cuenta en esta evaluación. En esta ocasión, este es un aspecto derivado de la calidad del software producido por cada modelo: si el software tiene alguna clase de deficiencia, es posible que un gusano la explote. Una buena prueba para comprobar el impacto de esta clase de ataques la constituye la lista de trabajos actuales del US-CERT [6] - United States Computer Emergency Readiness Team. Actualmente (Fri, May 14 2004) la lista de incidencias está compuesta de:

- W32/Sasser
- Exploit for Microsoft PCT vulnerability released
- Exploitation of Outlook Express MHTML cross-domain scripting vulnerability
- Sober.F malicious code
- Exploit for Cisco vulnerabilities released
- Phatbot Trojan
- Many variants of W32/Beagle malicious code
- Many variants of W32/Netsky malicious code
- Many variants of W32/MyDoom malicious code

Todas ellas son vulnerabilidades de programas desarrollados con el modelo cerrado, aunque caben algunos posibles argumentos para este hecho. Seis de las nueve incidencias de seguridad son virus de Microsoft Windows; es posible que este hecho se deba a que existen muchos más desktops ejecutando este sistema operativo cerrado. Parece lógico pensar que los escritores de esta clase de programas se centrarán en la plataforma que domine el mercardo, para que sus creaciones tengan la mayor repercusión posible. También sería posible añadir los hechos anteriormente expuestos: si el software cerrado es más vulnerable por su forma de desarrollo y/o su mantenimiento, es lógico que existan más programas maliciosos que se aprovechen de esta deficiencia.

## V. SERVIDORES WEB

Por último, los servidores web serán la última herramienta software que se estudiará para comparar la seguridad del Software Libre y Cerrado.

TABLA VI
NETCRAFT: SERVIDOR WEB, MAYO 2004

| Servidor Web | Número | Portcentaje |
|---|---|---|
| Apache | 33,892,817 | 67,05 % |
| IIS | 10,858,168 | 21,48 % |
| SunONE | 1,644,412 | 3,25 % |
| Zeus | 763,302 | 1,49 % |

Ahora habría que estudiar el número de incidencias que se producen en cada uno de ellos. Si se cruzan estos últimos datos con los de popularidad en servidores de Internet es posible obtener un valor sobre el porcentaje de incidencias. Hay que tener en cuenta que este valor no es completamente justo, ya que los sistemas con una base instalada más amplia tienen una posibilidad mayor de sufrir ataques: un atacante tiene una posibilidad mayor de encontrar uno mal configurado o

vulnerable, existe un background mayor sobre la tecnología, etc.

Un estudio de Attrition[7] a este respecto muestra los porcentajes de ataque que sufrieron los servidores ordenados por el sistema operativo que ejecutaban.

TABLA VII
ATTRITION: ATAQUES EXITOSOS A SERVIDORES WEB

| Sistema Operativo | Porcentaje |
|---|---|
| Windows | 66,09 % |
| GNU/Linux | 17,01 % |
| Solaris | 8 % |
| *BSD | 6 % |

De los datos del estudio de Attrition se sacan varias conclusiones. En primer lugar se ve claramente que Microsoft Windows es el sistema más vulnerable a los ataques Web. En lo relativo a los sistema operativos libres, se puede observar como GNU/Linux sufre más ataques exitosos que la familia de los BSD; siendo todos ellos libres, parece que existe un mayor nivel de seguridad en unos que en otros.

Arthur Wong, CEO de SecurityFocus anunció en "RSA: Security in 2002 worse than 2001, exec says" [2] el resultado de las estadísticas de su compañía respecto a la seguridad de los servidores Web. Los datos presentan que IIS fué atacado 1400 veces más que Apache: mientras que el servidor de Microsoft había sido atacado 17 millones de veces, Apache lo había sido 12,000. Estos números tienen una especial relevancia si se tiene en cuenta que hay aproximadamente tres veces más servidores Apache que ISS en Internet. El número total de ataques de toda clase de sistemas en 2001 fueron 29 millones.

## VI. CONCLUSIONES

No es sencillo extraer una única conclusión de todo lo expuesto anteriormente. A lo largo del artículo se han tratado los posibles factores que intervienen en la mayor o menor seguridad del Software Libre respecto al Software Cerrado. Basándose en los datos expuestos, los sistemas basados en Software Libre son claramente más seguros. Las razones para este fenómeno podrían ser varias: es posible que sea por la motivación con la que se desarrolla, por la forma de trabajo, por la gestión de las vulnerabilidades o incluso por algunos aspectos sociales. Tampoco es seguro que esta situación se vaya a perpetuar, es decir, no hay una razón clara por la que esta situación no pudiera cambiar en un futuro, aunque tampoco hay ningún motivo para suponer que se producirá tal cambio. El hecho es que, hoy en día, los sistemas basados en Software Libre tienen un menor porcentaje de ataques con éxito, su actualización frente a vulnerabilidades es más ágil, están ampliamente implantados en muchos sectores y su tendencia de todo este movimiento es el seguir creciendo.

## VII. AGRADECIMIENTOS


- A David A. Wheeler <dwheeler@dwheeler.com>. Su documento "*Why Open Source Software / Free Software (OSS/FS)?*" es excelente, y un punto de referencia para el análisis práctico del movimiento del Software Libre.

- A todos los que hacen el Software Libre posible.


## REFERENCIAS


[1] BugTraq, "Bugtraq: http://www.bugtraq.org," .
[2] 11th RSA Conference, "Security in 2002 worse than 2001," http://www.cnn.com/2002/TECH/internet/02/25/2002.security.idg/index.html, February 25 2002.
[3] Netcraft, "Netcraft survey: http://www.netcraft.com/survey/," .
[4] Security Portal, "Security portal, fixing sleep: http://web.archive.org/web/20010608142954/http://securityportal.com/cover/coverstory .
[5] CERT, "Cert trends: http://www.cert.org/present/cert-overview-trends/index.htm," .
[6] US-CERT, "Us-cert current activity: http://www.us-cert.gov/current/," .
[7] Attrition, "Attrition os graph: http://attrition.org/mirror/attrition/os-graphs.html," .

# Cómo colaborar en el proyecto de software libre KDE

Pedro Jurado Maqueda (melenas@kdehispano.org)

***Resumen*— El tema de esta ponencia es explicar las diferentes vías que existen actualmente para colaborar y ayudar en el proyecto de software libre KDE, que es un proyecto para realizar un sistema de escritorio (aplicaciones, bibliotecas de desarrollo y gestor de ventanas) libre, fácil de usar a la vez que potente y con las herramientas necesarias para poder desarrollar nueva aplicaciones.**

## I. Introducción

Mucha gente al empezar a introducirse en el mundo del software libre desea colaborar en el mismo pero no sabe por dónde empezar ni tampoco a quién dirigirse, otros simplemente ni se lo plantean pensando que colaborar en S.L. es sólo programar cuando en verdad es mucho más, enviar fallos, traducir, producir documentación, crear tutoriales, ayudar a otros usuarios etc... A continuación plantearemos los métodos y las direcciones a las que hay que dirigirse para empezar a colaborar en un proyecto concreto como KDE, empezando desde lo más simple y que requiera menos conocimientos, hasta terminar en lo más técnico y que necesite unos conocimientos más avanzados.

## II. Primer paso: Usa KDE e informa de fallos

Este paso que en principio pueda parecer tan obvio no lo es tanto, para empezar a ayudar en cualquier proyecto, es necesario que se conozca medianamente y haberlo utilizado habitualmente durante cierto tiempo, así, al usarlo, puedes detectar fallos y enviarlos a **http://bugs.kde.org**, con ello ya estarás ayudando al proyecto.

Además al usarlo ya estás realizando un proceso de promoción, ya que tus conocidos como amigos o familiares pueden ver al escritorio en acción y ver de lo que es capaz.

## III. Resolver dudas de usuarios

Todo el mundo alguna vez ha tenido dudas al utilizar KDE, es normal ya que KDE es un escritorio que aunque sencillo en su uso diario, tiene unas poderosas y versátiles herramientas que le permiten tanto automatizar tareas, como aumentar su rendimiento o simplemente cambiar su apariencia gráfica.

Así que otra tarea que puede hacer un usuario es ayudar a otros usuarios a resolver las dudas que se le presenten, no hace falta ser un usuario experto, pero sí al menos ser un usuario habitual de KDE.

Hay muchos foros de distros particulares y listas de correo de LUGs en donde la gente puede preguntar cosas relacionadas con KDE y en donde se puede contestar, pero hay dos sitios a donde los usuarios se dirigen preferentemente para resolver sus dudas sobre KDE

- Foros de KDE-Hispano: (**http://www.kdehispano.org/modules/newbb**) Los foros se dividen por distribuciones, lo cual hace que se puedan consultar las preguntas referente a los sistemas que se conozcan, aunque hay problemas que cuyas soluciones se pueden extrapolar a otras distribuciones directamente.
- Lista de correo de KDE-Hispano: (**https://listas.hispalinux.es/mailman/listinfo/kde-hispano**) Esta es una lista moderada, por lo tanto sólo los que estén suscritos a ella pueden enviar y responder, normalmente se considera de más alto nivel que los foros, ya que están suscritos veteranos usuarios y simpatizantes de KDE. Sin embargo, buscando en sus mensajes archivados se pueden encontrar respuestas a dudas que se planteen en los foros u otras listas.
- Foros de KDE (en inglés y otros idiomas): **http://http://kde-forum.org/** Foros sobre KDE pero en inglés.

## IV. Traducir

Ayudar a traducir KDE o cualquier otra aplicación de software libre, al contrario de lo que pueda pensar la gente, no es una tarea que deba llevar a cabo gente con un nivel de inglés alto, con certificados o varios años residiendo en un país angloparlante, se puede empezar con tan sólo los conocimientos básicos aprendidos durante la enseñanza obligatoria y a partir de ahí ir mejorando poco a poco. Además ha de tenerse en cuenta que la mayoría de los programas no están hechos por angloparlantes nativos, lo cual hace que usen un lenguaje técnico informático y fácil de traducir en la mayoría de los casos.



ISBN 84-88783-71-X©2004

KDE-es es el grupo de traducción que se encarga de coordinar las traducciones y ayudar en lo posible a la gente con dudas en sus traducciones. Para ello tiene una lista de correo moderada y cerrada en **http://www.kybs.de/mailman/listinfo/kde-es**.

Se pueden clasificar las traducciones en tres tipos que veremos a continuación:

- Traducción de aplicaciones: Sin duda alguna la forma más fácil de empezar, lo corto de las entradas a traducir y lo repetitivo de sus mensajes (load, save, play, etc...) hace que sea muy fácil traducir las interfaces
- Traducción de la documentación de aplicaciones: Para esta tarea hace falta algo más de conocimientos y de tiempo, ya que el nivel de inglés utilizado es algo mayor al emplearse giros lingüísticos o simplemente frases más elaboradas. Además el número de palabras a traducir en la documentación es mucho mayor que el que puede encontrarse en la interfaz del mismo programa.
- Traducción de otros artículos y documentos: Normalmente se trata de artículos y documentos relacionados con KDE pero no dentro del ”ámbito oficial”. Habitualmente son publicados en webs sobre software libre en general como Ars Technica (**http://arstechnica.com/**) u Osnews (**http://www.osnews.com**) y sus traducciones por norma general son publicados en KDE-Hispano. Estos mismos artículos son anunciados y comentados en KDE News (**http://dot.kde.org**) y es allí a donde nos podemos dirigir para ver que interesantes artículos hay publicados y dignos de ser traducidos

Para empezar a traducir interfaces y documentación de aplicaciones oficiales de KDE hace falta primero suscribirse a la lista de traductores por dos razones:

La primera es para que se le asigne trabajo y así no duplicar esfuerzos, para ello hay que ponerse en contacto con el coordinador (instrucciones en **http://es.i18n.kde.org**).

La segunda razón es para estar al tanto de las congelaciones de mensajes y plazos para tener las traducciones listas antes del plazo fijado, también para dar cuenta de errores encontrados o simplemente resolver dudas.

Aunque lo conveniente es primero apuntarse a la lista y estar un par de meses recibiendo y leyendo correos sin pedir aún ningún paquete para así ir acostumbrándose a la dinámica de grupo y al proceso mismo de traducir.

Para traducir documentos externos e incluso aplicaciones KDE que aún no han entrado de forma oficial no hace falta suscribirse a la lista de correo, aunque es conveniente por si surge alguna duda poder recibir ayuda de gente con experiencia y avisar también de que se va a traducir una aplicación para evitar duplicar esfuerzos, ya que si alguna vez pasa oficialmente el proyecto a KDE, se podría reutilizar lo ya traducido en vez de empezar de nuevo desde cero. Por otra parte también sería conveniente echar un vistazo a la guía de estilo de traducción del equipo español de traducción de KDE para intentar mantener la homogeneidad lo más posible.

No olvidemos que hay otra idiomas en el estado Español además del castellano, y que la mayoría de ellas tienen sus propios proyectos de traducción como son:

- Catalán: **http://ca.i18n.kde.org/**
- Gallego: **http://www.trasno.net/kde/**
- Vasco: **http://kdeuskaraz.euskalgnu.org/**

## V. Escribir artículos y tutoriales

Si eres un experto en algún programa y quieres compartir tus conocimientos puedes escribir un artículo o mini-tutorial sobre algún aspecto en concreto en KDE que no haya sido previamente explicado, ni en un artículo en español ni en ninguna otra lengua.

Dónde publicarlo es a elección del autor, pero un buen sitio para hacerlo sería **http://www.kdehispano.org**, aunque sirve cualquier web sobre software libre, o por qué no, cualquier revista en papel que paguen lo suficiente ;-), aunque se agradecería que se pusiera un aviso en la propia página web de kdehispano para poder tener así el mayor número de artículos sobre kde en español indexados,

Si se tiene un nivel de inglés lo suficientemente avanzado, también se podría traducir al inglés, así se alcanzaría a un número mayor de usuarios, no sólo porque en inglés puedan leerlo más gente, sino porque al ser el inglés la lengua «oficial» en el proyecto KDE, éste articulo podría ser traducido a otros idiomas por parte de otras personas.

## VI. Empaquetar aplicaciones para tu distribución favorita

Aunque empaquetar en un primer momento parece un proceso difícil largo y laborioso, en verdad no lo es tanto una vez que se pone manos a las obras y se trata de empaquetar programas simples con a lo sumo uno o dos binarios.

Tanto si se empaqueta paquetes deb (Debian, Guadalinex, Linex, etc...) como rpm (Red Hat, SuSE, Mandrake, etc...) o incluso tgz (Slackware), se puede utilizar un programa de fácil uso llamado checkinstall que a partir de los fuentes y en unos sencillos pasos compila y empaqueta el programa. No es una solución definitiva ni tampoco ”fina”de realizar este proceso, pero mejor algo que nada.

## VII. Metiendo mano al código fuente

Aunque programar en Qt no es difícil, puede costar algo de trabajo comenzar a entender todas las tecnologías que hay detrás de Qt y también de KDE que amplía a la primera, por ello, antes de lanzarnos a crear nuestra primera aplicación o ayudar en el desarrollo de una ya comenzado, podemos empezar por niveles más bajos

### *VII-A. Buscar y solucionar fallos*

Si en la primera parte del documento se explicaba como enviar fallos al proyecto KDE, podemos de igual forma y con los conocimientos mínimos necesarios resolver y solucionar esos fallos, para gente principiante KDE ha creado la categoría de fallos JJ (Junior Jobs). Son fallos de fácil solución que se marcan de esta manera para que los nuevos desarrolladores se animen y empiezen con retos de baja dificultad que hagan que poco a poco vayan cogiendo confianza y experiencia sin desanimarse por no encontrar la solución óptima, o simplemente una solución.

El conjunto de fallos es el siguiente
**http://bugs.kde.org/buglist.cgi?short_desc_type =allwordssubstrshort_desc=JJ %3Abug_status =UNCONFIRMEDbug_status=NEWbug_status =ASSIGNEDbug_status=REOPENED**

La forma de solucionar y enviar los fallos se hace mediante diff que envía sólo las diferencias entre el archivo original y el modificado para solucionar el fallo.

Pare crear este archivo diff sólo hace falta el comando *diff -u -p fuente.cpp.orig fuente.cpp > ./patch.diff*

donde fuente.cpp.orig es el archivo original que contiene el error y fuente.cpp es el archivo en donde el error está solucionado.

Entonces el archivo diff generado se puede enviar directamente al autor(es) directamente o bien se puede copiar y pegar en un comentario de bugs.kde.org

Una vez enviado el parche puede ocurrir varias cosas

- Que nadie conteste: en este caso quizás no lo hayas enviado a la lista o persona correcta, o no esté bien comentado, o quizás el desarrollador en cuestión no tiene tiempo, lo mejor en este caso es tratar de comentarlo mejor o tal vez enviarlo a alguna otra persona o lista que sea la correcta
- Que el parche cause conflicto con otro que se esté realizando, en este caso puedes ofrecer tu colaboración con el desarrollador
- Que no sea aceptado, puedes tratar de mejorarlo o bien discutir en la lista correspondiente por qué tu parche debe de ser aceptado.
- Que el desarrollador te pida unos cuantos cambios para ser aceptado, en este caso hazlos y si tienes alguna duda pregunta al desarrollador o en la lista de correo correspondiente
- Que el parche sea aceptado, en ese caso enhorabuena, has contribuido en uno de los mejores proyectos de software libre :-)

### *VII-B. ¿Qué es esto?*

Los tooltips o ayudas contextuales de ¿Qué es esto? son aquellos bocadillos que aparecen en forma de ayuda cuando seleccionamos Ayuda-¿¿Qué es esto? y después pulsamos con el ratón sobre el elemento del que tengamos dudas de su utilidad.

Para saber como introducirlos hay un interesante artículo de Aaron J. Seigo que nos explica de una forma sencilla y clara como hacerlo. No es la intención de esta ponencia explicar como, pero a modo de resumen sería tan fácil como introducir debajo del widget que queremos explicar la siguiente línea de código

**QWhatsThis::add( widget, i18n( ”Texto explicativo ”));**

No olvidando incluir los archivos de cabecera *klocale.h* y *qwhatsthist.h*

Más información en **http://urbanlizard.com/∼aseigo/whatsthis_tutorial/**

### *VII-C. Programar aplicaciones en KDE*

Digamos que este es el punto final para ayudar en el proyecto KDE, una vez que te decidas puedes hacer dos cosas

- Comenzar nuestra propia aplicación, si tienes alguna idea de un programa que todavía no existe en el software libre en general, o que aunque exista, no satisfaga todas tus necesidades y quieras darle otra orientación desde el principio, puedes empezar una nueva aplicación desde cero.
- También puedes unirte al desarrollo de una aplicación ya en desarrollo, o retomar el desarrollo de alguna aplicación abandonada.

Para programar aplicaciones en KDE puedes usar C++ para programar tus aplicaciones, aunque hay otras opciones, puedes consultar la lista de bindings disponibles para otros lenguajes en **http://developer.kde.org/language-bindings/**

De forma general se puede consultar los tutoriales, howtos y documentación existente para programar en KDE en la dirección **http://developer.kde.org/documentation/**, en

particular se recomienda el siguiente tutorial de Antonio Larrosa sobre como comenzar a programar para KDE/Qt **http://devel-home.kde.org/∼larrosa/es/tutorial/index.html**

## VIII. Final

Y eso es todo, esperamos que con este pequeño artículo, mucha gente se anime a colaborar en el proyecto KDE ahora que sabe donde y a quien dirigirse para empezar a trabajar.

A continuación todos los enlaces dados en el artículo y numerados.

~

1. http://bugs.kde.org
2. http://www.kdehispano.org/modules/newbb
3. https://listas.hispalinux.es/mailman/listinfo/kde-hispano
4. http://kde-forum.org/
5. http://www.kybs.de/mailman/listinfo/kde-es
6. http://arstechnica.com/
7. http://www.osnews.com
8. http://dot.kde.org
9. http://es.i18n.kde.org
10. http://ca.i18n.kde.org/
11. http://www.trasno.net/kde/
12. http://kdeuskaraz.euskalgnu.org/
13. http://www.kdehispano.org
14. http://bugs.kde.org/buglist.cgi?short_desc_type =allwordssubstrshort_desc=JJ%3Abug_status =UNCONFIRMEDbug_status=NEWbug_status =ASSIGNEDbug_status=REOPENED
15. http://urbanlizard.com/∼aseigo/whatsthis_tutorial/
16. http://developer.kde.org/language-bindings/
17. http://developer.kde.org/documentation/
18. http://devel-home.kde.org/∼larrosa/es/tutorial/index.html

# wxPython: Aplicaciones multiplataforma con Python

Arturo Fernández Montoro

***Resumen – wxPython*** **es un completo *tookit* para el desarrollo de aplicaciones multiplataforma con interfaz gráfica de usuario con el lenguaje *Python*. Su flexibilidad y potencia lo convierten en el adecuado para el desarrollo rápido de aplicaciones que deben correr en distintas plataformas. Está orientado a eventos y posee un rico conjunto de *widgets* para construir complejas interfaces de usuario.**

## I. Introducción

Uno de los principales aspectos que deben ser tenidos en cuenta cuando se plantea el desarrollo de aplicaciones que puedan ejecutarse en distintas plataformas es la interfaz gráfica de usuario. La razón es la heterogeneidad de los sistemas gráficos existentes en las distintas plataformas y sistemas operativos. Habitualmente el desarrallo de aplicaciones es totalmente dependiente de la plataforma dónde va a ejecutarse este software, lo que complica el desarrollo de aplicaciones multiplataforma. Por ejemplo, las aplicaciones para *MS Windows* suelen utilizar las librerías de MFC (*Microsoft Foundation Classes*) [5] para trabajar con la interfaz gráfica. Obviamente, estas librería son totalmente dependiente de este sistema operativo. Igualmente ocurre si desarrollamos para sistemas UNIX utilizando las librerías de *MOTIF* [4].

Con el objetivo de cubrir este hueco existen librerías que nos permiten el desarrollo de aplicaciones con independencia del sistema gráfico nativo de la plataforma.

Otro condicionante en el desarrollo de aplicaciones multiplataforma es la elección del lenguaje de programación. Debemos elegir un lenguaje que nos permita generar ejecutables para el mayor número de plataformas posibles. Los lenguajes de *scripting* nos facilitan en gran medida esta tarea, debido a que sólo es necesario disponer del intérprete para cada sistema operativo. Basta con escribir una sola vez el código. Dentro de este apartado destacamos el lenguaje *Python* [2] por varios aspectos: es libre, está orientado a objetos, existen intérpretes para UNIX, Linux, Windows y Mac, posee una extensa API y es fácil de aprender.

En este contexto aparece *wxPython* [1]. Se trata una librería *open source* para desarrollar aplicaciones multiplataforma con interfaz gráfica en *Python*. Nos ofrece un completo conjunto de *widget* y de métodos para desarrollar la interfaz de usuario e interactuar con ella.

Arturo Fernádez trabaja en la empresa Atos Origin SAE (arturo.fernandez@atosorigin.com).

ISBN 84-88783-71-X©2004

## II. wxWindows: el precedente

*wxPython* es en realidad un *porting* a *Python* del *toolkit* para el desarrollo de aplicaciones gráficas en C++ llamado *wxWindows* [3]. Debido a problemas legales en septiembre de 2.003 se cambió el nombre a *wxWidgets*, aunque a lo largo de este artículo seguiremos utilizando el nombre original. Este completo *framework* nos proporciona un conjunto de *widgets* habituales en la programación de interfaces gráficas como por ejemplo cajas de texto, botones, ventanas, listas desplegables, *checkboxes* y cuadros de diálogo. Asimismo nos facilita las rutinas necesarias para interactuar con los mismos, las cuales están basadas en el modelo de eventos. Además también incorpora rutinas para manejar sistemas de ficheros virtuales y soporte para *OpenGL*.

La idea del desarrollo de este componente surgió de Julian Smart que trabajaba en el *Artificial Intelligence Applications Institute* de la Universidad de Edinburgo, allá por el año 1992. Y surgió de la necesidad de desarrollar una aplicación que pudiera correr tanto en terminales *X-Windows* como en PC's con *MS Windows*. Dado que las herramientas comerciales existentes en aquel momento no cumplian los requisitos que necesitaba, decidió comenzar el desarrollo de una librería propia. Con el paso del tiempo y con la colaboración de otras personas, esta librería terminó por formar parte de un proyecto *Open Source* que actualmente es *wxWindows*. Actualmente, la licencia que posee es *open source* y se denomina “wxWindows license”. Básicamente es una licencia de tipo L-GPL, con la salvedad de que la licencia de *wxWindows* permite desarrollar trabajos derivados propietarios. Esta licencia fue aprobada por el organismo “Open Source Initiative”.

## III. wxPython

Utilizando el lenguaje *Python* nos encontramos con distintas opciones a la hora de elegir un *toolkit* gráfico: *pyQT* [8], *pyGTK* [9], *Tkinter* [10] o *wxPython*. Este último se distribuye como un módulo que podemos integrar fácilmente siguiendo la instalación común para módulos en *Python*. Dado que se trata de un *porting* del original *wxWindows*, la curva de aprendizaje será mínima si ya tenemos experiencia con el *toolkit* para C++. La API es prácticamente igual, salvando las diferencias sintácticas que existen entre ambos lenguajes. Gracias a esto podemos considerar a *wxPython* como una de las mejores opciones para desarrollar aplicaciones multiplataforma con GUI en *Python*.

La última versión estable de *wxPython* es la 2.4.2.4, aunque también está disponible la versión 2.5.2.8 que se considera como “inestable” y en desarrollo. Existen tanto versiones para *Python 2.2* como para *Python 2.3*. Asimismo en el sitio

oficial podemos encontrar distribuciones binarias para *MS Windows, Linux (Red Hat, Mandrake y Fedora)* y *Mac OS X* Obviamente, también podemos descargar el código fuente y compilarlo para nuestra plataforma específica. Tal y como se indica en este sitio, los usuarios de *Debian GNU/Linux* pueden instalar el paquete (*libwxgtk2.2-python)* que existe a tal efecto en los *mirrors* habituales del proyecto *Debian.*

A la hora de realizar la instalación debemos tener en cuenta que *wxPython* depende de las librerías *GTK+*. En el caso de Linux nos aseguraremos de tener instaladas estas librerías antes de instalar el paquete o realizar la compilación manual.

En el caso de *MS Windows* basta con ejecutar el ejecutable distribuido que contiene un asistente de instalación.

Finalmente veremos que el módulo ha quedado instalado en el *path* por defecto dónde se instalan los módulos listos para ser utilizados en *Python.* Por ejemplo, en el caso de *Debian* es el directorio */usr/lib/python2.2/-packages/wxPython.*

Comprobaremos la correcta instalación invocando desde la línea de comandos al intérprete de *Python* y ejecutando el siguiente comando:

*from wxPython import **

Si no hay ningún error, la instalación se ha llevado a cabo con éxito y podemos pasar a trabajar con este *toolkit.*

Una de las principales características de *wxPython* es el modelo de orientación a eventos que incluye. Como respuestas a los diferentes eventos debemos escribir rutinas para tratarlos. Es lo que se suele conocer con el nombre de *callback routines*, que también existen en otros lenguajes y sistemas para el desarrollo de GUI's. w*xPython* nos ofrece un completo conjunto de eventos que pueden ocurrir sobre los distintos elementos de la interfaz gráfica.

A continuación pasaremos a describir los principales objetos de la interfaz de usuario que podemos utilizar para nuestros desarrollos.

### *III -A. Principales widgets*

Cada uno de los elementos de la interfaz gráfica está implementado como una clase o un conjunto de estas. Este hecho nos garantiza la portabilidad entre plataformas. Pensando en la distintas categorías tenemos la siguiente clasificación:

- Ventanas *managed*: Dentro de esta categoría encontramos las ventanas que dependen de otras, por ejemplo cuadros de diálogo, *frames* MDI y diálogos asistentes (*wizards*).
- Ventanas comunes: Como son paneles, *frames* o *grids.*
- Controles genéricos: Imprescindibles en la interacción con el usuario. Tenemos botones (*wxButton*), *checkboxes* (*wxCheckBox*), listas desplegables (*wxComboBox*), etiquetas de texto estáticas (*wxStaticText*), cajas de texto para entrada de valores (*wxTextCtrl*), etc.
- Menús: Es posible construir menús añadiendo distintas opciones.
- Estructura y manejo de ventanas: Posicionamiento relativo y absoluto, tamaño de las mismas, etc.
- Controles gráficos: *wxPython* incorpora un completo conjunto de *widgets* para manejar imágenes gráficas dentro de ventanas y componentes asociados.

Observando la completa API encontraremos muchos más controles para satisfacer los requisitos de aplicaciones que requieren complejas interfaces gráficas. Una buena opción para tener una idea general de los distintos *widgets* que podemos utilizar, es ejecutar la aplicación de ejemplo que se distribuye como parte de la documentación.

### *III-B. Comenzando el desarrollo*

Una vez que sabemos sobre *wxPython*:

- Que está orientado a eventos.
- Los principales componentes que podemos utilizar.

Estamos en condiciones de realizar nuestra primera aplicación. Comenzaremos escribiendo una clase o un conjunto de ellas que representen un espacio virtual de la pantalla dónde posicionaremos los distintos *widgets.* El componente principal que nos ofrece *wxPython* se llama *wxWindow*. Sin embargo, para realizar una simple aplicación es más comodo utilizar una clase que hereda de esta y que implementa el comportamiento necesario para manejar ventanas, su nombre es *wxFrame* y es en realidad un contenedor de controles. Seguidamente debemos tener en cuenta la llamada que hemos de realizar al método *wxPySimpleApp()* para obtner la instancia que necesitamos. Por último debemos indicarle a esta instancia que debe estar preparada para manejar eventos, esto se hace con la llamada al método *MainLoop*(), el cual nos marca el ciclo inicial. Con estos simples pasos podemos desarrollar una aplicación sencilla. Como ejemplo, veremos el típico “Hola Mundo”. El código sería el siguiente:

```
from wxPython.wx import *
if __name__ == '__main__':
        app = wxPySimpleApp()
        frame = wxFrame(None, -1, 'Hola Mundo', size=(200,100))
        frame.Show(true)
        app.MainLoop()
```

Simplemente con estas líneas ya podemos ver una ventana en pantalla. Pensemos en el número de líneas de código que necesitaríamos para obtener el mismo resultado utilizando lenguajes como C o Java. La ventaja de *Python* es claramente significativa en este aspecto.

En la figura 1 podemos apreciar la ventana de nuestra primera aplicación corriendo en Linux, asimismo en la figura 2 vemos el resultado bajo *MS Windows*.

Evidentemente podemos desarrollar aplicaciones más complejas. En la figura 3 podemos ver un *front-end* desarrollado por el autor para el program *jhead*, el cual nos permite leer el formato *EXIF* [11] que poseen las fotografías digitales compatibles con este formato.

### III-C. Eventos y rutinas *callback*

Formalmente un evento es un mensaje enviado por el manejador de eventos para indicarle a la aplicación que algo ha ocurrido. La aplicación debe responder a este hecho. Es responsabilidad del desarrollador escribir el código de respuesta al evento, mientras que la captura del mismo se

realiza de forma transparente. Las rutinas que realizan la tarea de respuestas son conocidas con el nombre de *callbacks*.

Con el objetivo de facilitar la tarea de conexión entre eventos y respuestas, *wxPython* nos ofrece unos métodos especiales que comienzan con el prefijo *EVT*. Es conveniente revisar la API para ver cuales de estos métodos podemos utilizar como respuesta a los eventos que surgen sobre los *widgets*. A continuación podemos ver un ejemplo de método para controlar los eventos sobre un *wxComboBox*:

*EVT_COMBOBOX(self,self.cb.GetId(), self.onChangecb)*

El primer parámetro es el propio objeto, el segundo nos indica el identificador el control en cuestión y el


Figura 1. “Hola Mundo” corriendo en Linux con KDE.


Figura 2. “Hola Mundo” corriendo en Windows.

tercer parámetro referencia la rutina de *callbak*. Es en esta rutina dónde debemos escribir el código de respuesta cuando ocurra un evento sobre el *wxComboBox*, en este caso el evento hace referencia al cambio de valor sobre el *combox*

## IV. Conclusiones

*wxPython* representa una opción muy a tener en cuenta en el desarrollo de aplicaciones multiplataforma. Su amplio conjunto de *widgets* y el aval de estar basado en la estabilidad y experiencia de *wxWindows* lo convierten en una opción muy interesante. Si a esto unimos la flexibilidad y potencia del lenguaje *Python* estamos ante uno de los *toolkits* gráficos más atractivos de los últimos años. Otra gran ventaja es que utilizando Linux y *wxPython* tenemos un entorno para el desarrollo de aplicaciones totalmente libre, que es otra cuestión que muchos desarrolladores tienen en cuenta, debido a que la mayoría de entornos de desarrollo utilizan software propietario en mayor o menor medida.

Sin embargo, encontramos una carencia: no existe un IDE totalmente funcional para desarrollar con *wxPython*. Actualmente *BOA Constructor* [7] está en fase de desarrollo, aunque las versiones beta que existen prometen bastante. Esperemos que con el tiempo surgan otros IDE's que nos ayuden a convertir a *wxPython* en un estándar para el desarrollo rápido de aplicaciones multiplataforma.

Figura 3. Aplicación front-end para *jhead* desarrollada con wxPython.

## Referencias


[1]wxPython, http://www,wxpython.org
[2]Python, http://www.python.org
[3]wxWindows, http://www.wxwindows.org
[4]Motif, http://www.opengroup.org/motif/
[5]MFC,http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vcmfc98/html/mfchm.asp
[6]GTK, http://www.gtk.org
[7]Boa Constructor, http://boa-constructor.sourceforge.net
[8]PyQT, http://www.riverbankcomputing.co.uk/pyqt/index.php
[9]PyGTKhttp://www.pygtk.org
[10]TKinter, http://www.python.org/topics/tkinter/
[11]EXIF, http://www.exif.org

# CVSAnalY: una herramienta libre para el análisis de repositorios CVS

Israel Herraiz, Gregorio Robles, Jesús M. González-Barahona

***Resumen*— El desarrollo de software libre ha demostrado en los últimos años una clara tendencia de consolidación con varios proyectos calificados indiscutiblemente como grandes éxitos. Sin embargo, a pesar del creciente interés muchos aspectos relativos al software libre y a las comunidades que lo crean permanecen inexplorados y en ocasiones incomprendidos. La obtención de datos cualitativos y cuantitativos y su posterior análisis es, por tanto, de gran importancia. En este artículo presentamos una herramienta que permite realizar un análisis remoto y desatendido de proyectos de software libre, y cuyos resultados son la base para la evaluación ingenieril, social y económica de estos proyectos. Esta herramienta, denominada CVSAnalY, puede encontrarse en** `http://libresoft.dat.escet.urjc.es/cvsanal`**. Nuestro grupo de investigación emplea, entre otras, esta herramienta para la caracterización y análisis de proyectos de software libre, dentro de la acción coordinada CALIBRE, impulsada por la Comisión Europea (**`http://calibre.ie`**).**

## I. Introducción

Una de las principales características de los proyectos de software libre es la dispersión. No existe una autoridad central encargada del proyecto a la que se pueda acudir para solicitar información acerca del mismo. Además, los usuarios y desarrolladores están dispersos geográficamente. Por tanto, la dispersión de los proyectos de software libre es tanto administrativa como geográfica[1].

Si comparamos esta situación con la que se enfrentaba la ingeniería del software tradicional veremos que es todavía más complicada de analizar. Al fin y al cabo, en los ámbitos tradicionales se conoce tanto el número de desarrolladores que están asignados a un proyecto software dado, así como la dedicación (generalmente medida en horas) que éstos dedicaban al proyecto. En el software libre, en general, carecemos de esta información, pero tenemos a nuestra disposición una serie de datos que nos permiten indagar profundamente en los mismos. Éste es precisamente uno de los objetivos del proyecto CALIBRE, una acción coordinada de la Comisión Europea cuyo objetivo es acercar el desarrollo y las metodologías utilizadas en el software libre a las industrias europeas de mayor importancia como son la de la automoción, las telecomunicaciones, etc.

Gran parte de los datos que podemos obtener, están disponibles públicamente a través de Internet, la tecnología que ha permitido evitar los problemas de la dispersión de los proyectos de software libre. Entre las herramientas utilizadas, nos encontamos con los sistemas de control de versiones (como el *Concurrent Version System*, CVS), que suponen una herramienta de trabajo coordinado y centralizado. Estos sistemas permiten trabajar sobre el mismo proyecto a varias personas de manera remota, y registrar todas las operaciones realizadas por los desarrolladores sobre el proyecto. Además, los usuarios pueden obtener la última versión (o cualquier versión anterior) desde el CVS.

Otra característica muy interesante de estos sistemas, es que no sólo se guardan registros de las operaciones y código fuente de la última versión del proyecto, si no de todas las versiones anteriores, lo que permite acceder a cualquier momento del desarrollo del proyecto, o realizar análisis históricos del proyecto.

Debido a estas características, los sistemas de control de versiones son el campo de exploración perfecto para analizar el desarrollo de un proyecto de software[1].

En este artículo se muestra una herramienta que emplea la información almacenada en un repositorio CVS, para realizar un análisis del desarrollo de un proyecto de software, de una manera no intrusiva.

## II. ¿Qué hace CVSAnalY?

CVSAnalY[2] obtiene la mayor parte de la información acerca del proyecto de software libre de los *logs* (o históricos). Cada vez que un desarrollador realiza un cambio en el CVS (como por ejemplo, añadir nuevo código al proyecto) se registra el nombre del usuario que realizó el cambio, la fecha, qué líneas ha añadido

Universidad Rey Juan Carlos. Grupo de Sistemas y Comunicaciones.

{herraiz, grex, jgb}@gsyc.escet.urjc.es

[1]Aunque en general, sólo los proyectos de software libre tienen su código fuente almacenado en un CVS de manera pública.

ISBN 84-88783-71-X©2004

y cuáles ha eliminado, qué ficheros ha añadido y/o eliminado, etc. Además, el desarrollador también puede añadir un comentario en el cambio.

CVSAnalY extrae esta información de los logs, la procesa, la inserta en una base de datos y después calcula diversos parámetros estadísticos. Los pasos principales de ejecución son:

1. Preproceso:
   *a*) Descarga las fuentes del proyecto.
   *b*) Descarga los logs.
   *c*) Extrae la información de los logs.
   *d*) Identifica el tipo al que corresponde cada fichero (documentación, código, imágenes, traducciones, interfaz de usuario y sonidos).
   *e*) Contabiliza el número de líneas de los ficheros de código, empleando la herramienta `wc`.
2. Inserta la información extraída y procesada en una base de datos. También existe la posiblidad de que sea exportada a XML.
3. Cálculos estadísticos. Calcula diversos índices acerca de la igualdad en la contribución al proyecto por parte de los desarrolladores y los representa gráficamente. También puede dibujar gráficas con la evolución a lo largo del tiempo del proyecto.

Además, los resultados pueden exportarse a un servidor web. CVSAnalY crea una interfaz web que permite explorar los resultados de una manera muy sencilla. Podemos encontrar varios ejemplos de estas webs en la página principal de CVSAnalY (ver Resumen). Por ejemplo, en la web de resultados podemos obtener una lista de desarrolladores, elegir uno y obtener, entre otra información, el número de líneas que ha añadido al proyecto a lo largo del tiempo.

De una manera muy general, los resultados que hay disponibles son:

- Número de módulos, *commits*, *committers*[2], número total de líneas de código, número de líneas añadidas o eliminadas, etc
- También históricos de estos datos (por ejemplo, número de modulos en este momento, hace un año, hace dos, etc).
- Indicadores de desigualdad (para comprobar, por ejemplo, si una fracción pequeña de todos los desarrolladores contribuye con la mayoría del código del proyecto).
- *Generaciones* de desarrolladores. En este aspecto nos detendremos en un punto posterior, puesto que es una conclusión muy interesante acerca de los proyectos de software libre, que hemos podido identificar gracias a CVSAnalY.

[2]Por *commit* entendemos cualquier cambio realizado en el proyecto, y por *commiter* la persona que realiza este cambio en el CVS. No necesariamente tienen que ser cambios en el código del proyecto, sino que pueden ser en cualquier aspecto del mismo: documentación, imágenes, traducciones, etc.

## III. Cómo obtener los resultados

CVSAnalY se encuentra en una fase muy activa de desarrollo, y casi cada día se realizan modificaciones y se añaden nuevas características a la herramienta. En este momento, el único modo de emplear la herramienta es editando el fichero de configuración `config.py`. Entre otros datos, se necesita la dirección del CVS, la lista de módulos que se van a descargar y analizar (se puede indicar que se analicen todos los módulos de una manera sencilla), el nombre de usuario para acceder al CVS, qué análisis se van a realizar, qué gráficas se obtienen, dónde se guardarán los resultados, a qué servidor web se exportarán, etc.

Este fichero es ciertamente complejo. Para paliar la dificultad de configurar todas las opciones correctamente, junto con la herramienta se distribuyen ficheros de configuración adaptados a los proyectos de software libre más conocidos (Apache, GNOME, KDE, GCC, etc). De este modo, se puede emplear una de estas preconfiguraciones como punto de partida para el análisis de un proyecto.

En el caso de que no se especifique alguna opción de configuración imprescindible, la herramienta muestra un menú que nos permite seleccionar el valor adecuado para esa opción. Además, si el valor es simplemente verdadero o falso, podemos seleccionar que para cualquier otra incidencia próxima se tome un valor por defecto, y no se pregunte al usuario qué valor debe tomarse para la opción perdida.

Para emplear CVSAnalY existen una serie de prerrequisitos. A continuación mostramos la lista de paquetes Debian necesarios para poder ejecutar el programa[3]: cvs, mysql-server, python (versión 2.2 o posterior), python-mysql, python-mysqldb, python-imaging, gnuplot, proticus (versión 2.2 para obtener diagramas de sectores en color), smail, R, whirlgif, rsync (sólo si se va a usar el rsync del repositorio cvs), cron, rcs (sólo si se emplea rsync).

Una vez completado el fichero de configuración, e instalados todos los paquetes necesarios, para ejecutar la herramienta simplemente hay que escribir `python cvsanaly.py` dentro del directorio donde tengamos instalada la herramienta.

La ejecución puede llevar mucho tiempo, sobre todo si estamos analizando proyectos muy grandes. Además, se

[3]En otros sistemas los nombres de los paquetes son prácticamente iguales. Por ejemplo, nosotros hemos probado la herramienta en FreeBSD y los paquetes necesarios eran prácticamente los mismos.

realizan muchas peticiones al CVS, por lo que no debemos realizar muchos análisis continuados sobre el mismo CVS. En cualquier caso, para probar la herramienta y familiarizarse con ella es mejor usar repositorios CVS de proyectos pequeños. Por ejemplo, en SourceForge (`http://sf.net`) podemos encontrar muchos proyectos con CVS no demasiado grandes.

## IV. Ejemplos de resultados obtenidos

La cantidad de información que se obtiene como resultado de un análisis es inmensa. Por suerte, mediante la interfaz web se obtienen clasificados, y pueden consultarse de una manera sencilla. Entre otros resultados se obtienen diversas gráficas para poder entender mejor el significado de los resultados.

En este punto tan sólo vamos a exponer algunos ejemplos de las gráficas que se obtienen, explicando qué información proporcionan.

### *IV-A. Evolución del número de líneas de código*

Por ejemplo, vamos a mostrar la evolución en el número de líneas de código de Evolution, un cliente de correo electrónico y agenda del escritorio GNOME. La gráfica obtenida con CVSAnalY se muestra en la figura 1.

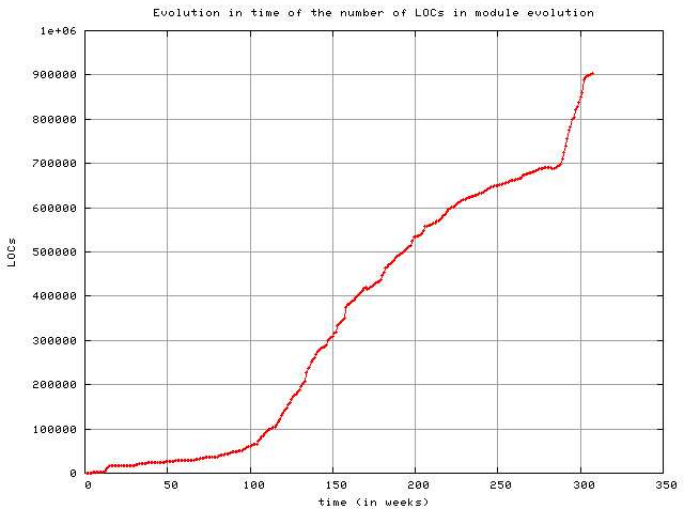



Fig. 1. Evolución del número de líneas de código de Evolution

En este caso, la gráfica nos ha permitido identificar un momento clave en el desarrollo de Evolution. Alrededor de la semana 100 se puede observar un cambio de pendiente en la gráfica. Ese punto se ha identificado como el momento en el que Ximian se hizo cargo del proyecto. Se observa claramente cómo Ximian logró aumentar la velocidad de desarrollo del proyecto.

Los datos mostrados pueden emplearse también para realizar estimaciones del coste de desarrollo, empleando modelos como COCOMO[3].

### *IV-B. Indicadores de desigualdad en las contribuciones al proyecto*

Otro parámetro interesante que se puede obtener es el coeficiente de Gini[4]. Este coeficiente es un indicador de desigualdad; mide si la mayoría de las aportaciones al proyecto se deben a una minoría de desarrolladores. En la figura 2 se muestra la gráfica para el caso de Evolution.

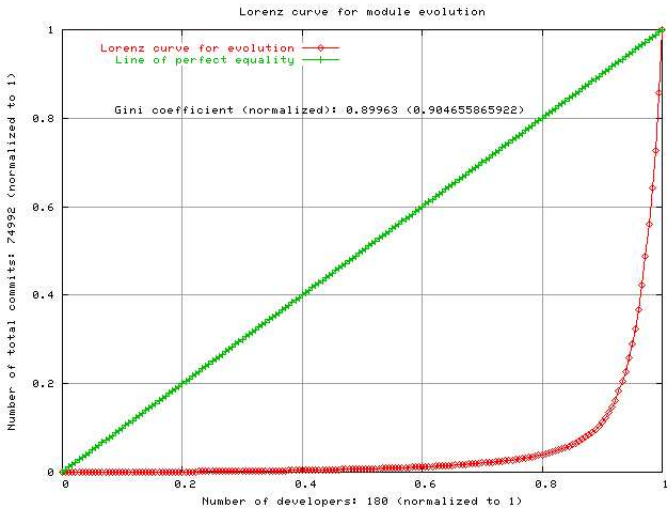



Fig. 2. Indicador de desigualdad para las aportaciones a Evolution

La línea recta indica igualdad. Esto es, todos los desarrolladores contribuyen en la misma cantidad al proyecto. El coeficiente de Gini mide la relación entre las áreas bajo la recta y la curva que muestra la distribución de aportaciones en función de los desarrolladores. Cuanto más cerca esté esa relación de 1, más desigual ha sido la participación en el proyecto. En general, en proyectos de software libre nos encontramos con valores que rondan el valor 0.8, por lo que una minoría de programadores contribuye a la mayoría del desarrollo (es decir, se verifica el principio de Pareto[4] aplicado al desarrollo del proyecto).

## V. Generaciones de desarrolladores

Es una idea común que el desarrollo del software libre avanza gracias a la aportación de individuos clave. Así, muchos piensan que si no existiera Linus Torvalds nunca se hubiera logrado desarrollar el kernel de Linux. O que Miguel de Icaza es un desarrollador imprescindible para el éxito del escritorio GNOME.

[4]Habitualmente este principio se enuncia así: *el* 20 % *de las causas originan el* 80 % *de los problemas.*

Estos desarrolladores, normalmente denominados *code gods*, son, en su mayoría, los que más contribuyen al desarrollo de los proyectos en los que participan. Sin embargo, nuestros estudios de varios proyectos muestran que no existen desarrolladores imprescindibles en el software libre[5]. En cambio, el desarrollo tiene varias etapas, entre las cuales existe un relevo generacional.

En la figura 3 se muestra una representación gráfica de las generaciones de desarrolladores. El desarrollo del proyecto se ha dividido en 10 intervalos iguales. Se identificó el grupo de desarrolladores más activo en cada intervalo, y se representó la contribución de cada grupo durante todo el proyecto.

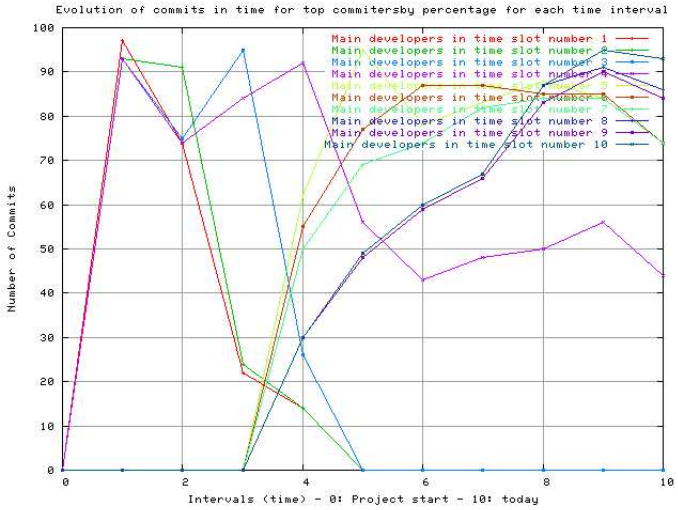



Fig. 3. *Generaciones* de desarrolladores en Evolution.

Si nos fijamos por ejemplo en el grupo más activo en el primer intervalo, vemos que en ese período de tiempo realizaban casi todas las contribuciones al proyecto. En cambio, este mismo grupo dejó de contribuir al proyecto en el quinto intervalo. Y durante ese período de tiempo, sus contribuciones fueron disminuyendo. El grupo más activo en ese quinto intervalo no aportaba nada al proyecto en el primer intervalo, y como se observa son los principales desarrolladores en intervalos posteriores.

Por tanto podemos concluir que existen generaciones de desarrolladores. Los desarrolladores más activos en el comienzo del proyecto ya no están participando en su desarrollo, y han sido relevados por otro grupo de desarrolladores que ahora aportan la mayoría de las contribuciones. Es decir, existe un relevo generacional. Este comportamiento se ha observado también en otros proyectos de software libre.

El tratamiento de los datos necesarios para obtener esta representación gráfica se realizó de manera automatizada con CVSAnalY, gracias a que en el CVS se guarda el estado del proyecto en todos los instantes en los que se han producido cambios, y además se guarda también quién es el autor de cada cambio. Por tanto fue posible trazar todo el trabajo de los desarrolladores en todo el período de existencia del proyecto.

## VI. Conclusiones

CVSAnalY es una herramienta para analizar proyectos de software libre, de una manera remota y desatendida. Permite obtener una perspectiva global del proyecto, desde su comienzo hasta la actualidad.

Tradicionalmente se han realizado muchas suposiciones acerca del software libre[6], y no se han fundamentado en datos o estudios empíricos. Todos estos trabajos no hubieran sido aceptados en cualquier otra disciplina científica, por no ser verificables. Por tanto, es el momento de realizar análisis rigurosos y fundamentados en datos experimentales del software libre (por ejemplo, el estudio [7]), que nos permitan extraer conclusiones válidas acerca del mismo.

Además, los resultados obtenidos son la base para una evaluación económica de los proyectos de software libre, lo que permitiría introducir los modelos abiertos de desarrollo dentro de la industria. Otro punto muy importante de los resultados de estos análisis es el de los recursos humanos, puesto que es posible trazar el trabajo de cada desarrollador desde el inicio del proyecto hasta el momento actual (lo que resultaría muy útil en un entorno empresarial).

Sin embargo, CVSAnalY tiene que seguir mejorando. Un verdadero estudio de un proyecto de software libre no debería basarse en el estudio de aspectos técnicos. La organización social formada por los desarrolladores y los usuarios es muy importante, y no debería obviarse[1]. Por tanto sería necesario también estudiar por ejemplo las listas de correo del proyecto[8]. Sin olvidarnos de los sistemas de seguimiento de fallos, que también pueden aportar información muy valiosa acerca de la evolución del proyecto. Es decir, debería realizarse un análisis integral del proyecto de software libre[9].

Además, para facilitar su uso, necesitaría una interfaz gráfica de usuario amigable, que permitiera seleccionar la configuración adecuada mediante asistentes. También sería necesario escribir una documentación completa, sobre todo de las múltiples opciones de configuración que posee, y una lista de preguntas frecuentes explicando en qué consiste cada resultado obtenido, y, quizás, proporcionando enlaces a artículos de interés relacionados con los cálculos realizados.

Otro punto de desarrollo futuro consiste en discriminar entre lenguajes a la hora de contabilizar las líneas de código. En este momento se identifica el tipo de fichero

que existe en el CVS (sonido, código, traducciones, etc), pero no es distingue entre diferentes lenguajes de programación. Para ello sería interesante integrar la herramienta SlocCount[10] en CVSAnalY.

## REFERENCIAS


[1] Yuwan Ke; Kumiyo Nakajoki; Yasuhiro Yamamoto; Kouichi Kishida, "The co-evolution of systems and communitties in free and open source software development," in *Free/Open Source software development*, Stefan Koch, Ed. 2005, Idea Group Publishing.

[2] Gregorio Robles; Stefan Koch; Jesus M. Gonzalez-Barahona, "Remote analysis and measuremente of libre software systems by means of the CVSAanalY tool," in *Proceedings of the 2nd ICSE Workshop on Remote Analysis and Measurement of Software Systems (RAMSS '04). 26th International Conference on Software Engineering*, May 2004.

[3] Barry W. Boehm, *Software engineering economics*, Prentice Hall, 1981.

[4] Conrado Gini, *On the Measure of Concentration with Espacial Reference to Income and Wealth*, Cowles Commission, 1936.

[5] Jesús M. González-Barahona; Gregorio Robles, "Unmounting the code gods assumptions," Tech. Rep., Grupo de Sistemas y Comunicaciones. Universidad Rey Juan Carlos, 2003.

[6] Eric S. Raymond, "The cathedral and the bazaar," *First Monday*, 1997, http://www.firstmonday.dk/issue3_3/raymond/.

[7] Jesús M. González-Barahona; Gregorio Robles; Miguel Ortuño-Pérez; Luis Rodero-Merino; José Centeno-González; Vicente Matellán-Olivera; Eva Castro-Barbero; Pedro de-las Heras-Quirós, "Anatomy of two GNU/Linux distributions," in *Free/Open Source software development*, Stefan Koch, Ed. 2005, Idea Group Publishing.

[8] Giovano Francesco; Michèle Morner, "The knowledge ecology of open source projects," in *19th European Group of Organizational Studies Colloquium*, July 2003.

[9] Gregorio Robles; Jesús M. González-Barahona; Rishab Aiyer Gosh, "GlueTheos: Automating the retrieval and analysis of data from publicly available repositories," in *Mining Software Repositories Workshop. 26th International Conference on Software Engineering*, May 2004.

[10] David Wheeler, "SlocCount," http://www.dwheeler.com/sloccount.

# Proyectos fin de carrera y software libre

Antonio Araúzo Azofra, Antonio J. Cubero Atienza, Lorenzo Salas Morera

***Resumen* – El software libre, cada vez más presente en nuestra sociedad, está adquiriendo una importancia creciente en la Universidad por numerosas razones. Por otra parte, el proyecto fin de carrera es una materia muy importante en las carreras de ingeniería. Este trabajo analiza las relaciones entre ambos, y las posibles ventajas de su combinación, desde el simple uso de software libre en el proyecto fin de carrera, pasando por la liberación del proyecto, hasta la integración del proyecto en la comunidad del software libre y la elección de temas directamente relacionados.**

## I. Introducción

El Proyecto Fin de Carrera (PFC) es probablemente una de las materias a las que más importancia se da al cursar una carrera académica de ingeniería. Se trata de un ejercicio integrador o de síntesis de los conocimientos adquiridos durante toda la carrera, donde se profundiza en algún aspecto concreto o se desarrolla una aplicación definida. Por tanto, es quizá la materia más cercana a la posterior labor profesional.

Los proyectos fin de carrera pueden ser trabajos de ingeniería: de ejecución, de desarrollo o de investigación aplicada; requieren un trabajo considerable; y son realizados por alumnos que son casi ingenieros, pues les queda poco para terminar su carrera. Además, normalmente se trata de trabajos dirigidos o tutelados por profesores de universidad, y a veces en contacto con profesionales. Son, por tanto, unos trabajos de una envergadura y nivel de calidad considerables.

En general, todas las ingenierías incluyen en sus planes de estudios las asignaturas de Proyectos y Proyecto Fin de Carrera. Aunque los planes de estudios de las ingenierías en informática no las incluyen como materias troncales [1], las diferentes Escuelas sí suelen incluir el Proyecto Fin de Carrera (sobre todo en el caso de la ingeniería superior) y algunas la asignatura de Proyectos [2, 3].

Una situación habitual, por desgracia, es que el PFC después de presentado se quede en el olvido. Aunque en algunos casos los proyectos son utilizados para alguna investigación puntual, o en las clases de alguna asignatura, no cabe duda de que podrían ser todavía más útiles para más gente.

Por otra parte, el Software Libre (SL) es aquel en el que su/s autor/es otorgan licencia para ejercer las siguientes cuatro libertades:

(0) libertad de usarlo, sin ninguna restricción o discriminación, ni económica ni de cualquier otro tipo.
(1) libertad de estudiar cómo funciona, aprender de él, y adaptarlo (el acceso al código fuente es un requisito previo).
(2) libertad de distribuir copias, y así ayudar al prójimo.
(3) libertad de mejorar el programa y hacer públicas las mejoras a los demás, de forma que toda la comunidad se beneficie.

Existe una gran cantidad de personas, empresas y organizaciones, que usan, colaboran y/o desarrollan SL. Entre ellos existe un clima de cooperación infundido por la naturaleza del SL. Nos solemos referir al conjunto de todos ellos como la comunidad del SL.

Las Universidades están apoyando cada vez más el SL, y no son pocas las razones [4] para hacerlo. Hay un anillo web universitario pro SL, con el propósito de ser punto de encuentro de todas las asociaciones que comparten este fin. Además de que cada vez es mayor la implicación de profesores y alumnos, varias universidades han creado Oficinas de SL. Concretamente la Oficina de Software Libre de la Universidad de Cádiz (OSLUCA) ha expresado su interés en promover PFC relacionados con el software libre.

La simbiosis entre los proyectos fin de carrera y el software libre puede generar sinergias positivas muy interesantes para ambos. Un PFC, en vez de permanecer olvidado en una biblioteca, puede ser muy útil para el SL y la Sociedad en general, y a su vez un PFC puede tener grandes ventajas al sumergirse en el mundo del SL. Podría pensarse que esto es así sólo para informáticos, pero en realidad otras titulaciones también pueden aprovechar las ventajas de realizar un PFC relacionado con SL.

En este trabajo trataremos de mostrar las posibilidades que existen, y las ventajas que se pueden obtener, en la combinación de SL y los PFC. Partiremos desde el mero uso de SL en el desarrollo del proyecto, pasando por los detalles de cómo hacer libre un PFC, para terminar comentando cómo se puede lograr la integración completa de un PFC en el mundo y la comunidad del software libre.

## II. Uso de software libre en el proyecto fin de carrera

Existe actualmente tal cantidad de software libre que casi cualquier tarea que imaginemos posible realizar con un ordenador puede hacerse con algún programa libre, desde escribir una simple carta, pasando, por ejemplo, por el dibujo de moléculas químicas, hasta la realización de efectos especiales, como los de la película *Titanic*. Es cierto que

1 Aunque la materia Proyectos de ingeniería no está incluida como tal, sí están parte de sus contenidos dentro de la materia Sistemas Informáticos, que incluye como descriptor Proyectos de Sistemas Informáticos.

Los autores pertenecen al Área de Proyectos de Ingeniería del Departamento de Ingeniería Rural de la Universidad de Córdoba. (Email: arauzo (arroba) uco.es)

ISBN 84-88783-71-X©2004

existen algunas lagunas, como el reconocimiento automático de caracteres, pero casi siempre hay alguien que está trabajando por eliminarlas. Así, es muy probable que dentro de poco el software libre esté a la cabeza de la innovación en todos los ámbitos.

El uso de SL para el desarrollo de un PFC presenta diversas ventajas. La primera es poder usarlo sin tener que adquirir costosas licencias, "pactar con el diablo" (a veces regalan licencias con fines educativos, pero antes de explotar profesionalmente el proyecto habrá que pasar por caja), ni hacer nada ilegal. También hay programas libres que permiten realizar tareas que no se pueden hacer con programas no libres, o que funcionan mejor que otro no libre. Sin olvidar que incluso podemos adaptar el programa a nuestros intereses. A veces un pequeño cambio puede resultarnos muy útil. Además, colaborar con el SL es bueno para la Sociedad, y simplemente con usarlo ya se está colaborando con él. Porque aquel que lo usa probablemente hablará de él ayudando a difundirlo, enseñará a otros a usarlo, y si encuentra algún fallo o tiene alguna sugerencia puede comunicárselo a los desarrolladores para que lo mejoren.

Como indican en su presentación [5] los miembros del grupo de usuarios de GNU/Linux de la Universidad Carlos III de Madrid, hay muchos programas de SL que pueden ser de interés a la hora de desarrollar un PFC.

La redacción de la memoria puede hacerse con el Writer de OpenOffice, o algún otro procesador de textos libre como Abiword o Kword. LateX es algo mas difícil de usar pero se trata de una herramienta profesional de tipografía, que incorpora un soporte excelente de expresiones matemáticas y una gestión de citas bibliográficas muy versátil. Docbook es una alternativa más moderna a LateX para documentación técnica, pero quizá le falta un editor visual y mejorar la gestión de bibliografías.

Para preparar la presentación del proyecto disponemos, entre otros programas, del Impress de OpenOffice, muy completo y fácil de manejar, MagicPoint, que representa una forma distinta de preparar presentaciones, y también LateX puede ser usado para preparar presentaciones.

A la hora de programar disponemos de compiladores e intérpretes para diversos lenguajes (C, C++, Java, Fortran...). Queremos mencionar aquí especialmente a Python, un lenguaje de programación que es completamente libre (al contrario que otros como C++ donde la especificación de sus bibliotecas estándares no es pública, o Java donde éstas son públicas con restricciones), y además goza de un buen diseño, facilidad de aprendizaje y de uso, y una gran mantenibilidad del código. También hay editores con características que facilitan mucho la programación como X/Emacs, y diversos entornos completos de programación.

Podemos encontrar herramientas para trabajo en grupo que pueden ser útiles si el proyecto se desarrolla entre varios alumnos o en colaboración con otros. Por ejemplo, Subversion (SVN), una mejora del conocido CVS, permite trabajar a varias personas sobre los mismos ficheros simultáneamente y llevar un registro de los cambios.

El lenguaje R, y el programa GNUPlot nos permiten generar gráficos y estadísticas de datos que necesitemos analizar. También hay programas para simulación, gestores de contenidos web, y mucho software específico que puede estar relacionado con el tema concreto del proyecto. Por supuesto, debemos incluir los que permiten realizar otras tareas habituales, que pueden ser útiles o incluso necesarias para el desarrollo del PFC: navegar por la web (Mozilla, Galeon, Konqueror, w3m...), email (Mozilla Mail, Evolution, Mutt...), mensajería instantánea (Xchat, Gabber, Gaim, Amsn...), retoque de imágenes (Gimp)...

## III. Liberar un proyecto fin de carrera

Los derechos sobre el PFC, así como los de cualquier trabajo que realice un alumno en el desarrollo de su estudio, pertenecen en principio al alumno como autor de los mismos, según señala la Ley de Propiedad Intelectual [6] en su artículo 1 (el ámbito de esta ley incluye tanto la documentación, como el software, mencionados explícitamente en el artículo 10). Cuando para desarrollar el proyecto se cuente con financiación ajena, por ejemplo de un departamento de investigación universitario, tales como ayudas, becas o contratos, habrá que mirar las condiciones de los mismos para ver si éstas exigen la cesión, ya sea parcial o total, de los derechos. Si el proyecto es dirigido, es posible que parte de los derechos de autor correspondan a los directores, y por tanto quizá pertenezcan a la entidad donde éstos trabajen.

Las bibliotecas de las Escuelas de Ingeniería normalmente guardan copias de los PFC, y permiten su consulta bajo determinadas condiciones. Se considera que esta consulta puede ser útil para la comunidad universitaria, además de ser una garantía de transparencia. En estos casos la normativa de la universidad suele exigir al alumno varias copias del PFC y su consentimiento a que estén en la biblioteca como requisito previo a la presentación del proyecto. Se trata de una licencia o permiso del autor para que la biblioteca muestre su PFC, no de una cesión de los derechos de autor, que seguirán perteneciendo por completo al alumno.

Si el alumno posee los derechos de autor de su proyecto, puede otorgar licencias de uso como él crea conveniente. De esta forma, puede asignar a su proyecto una licencia, otorgada a todo el mundo, que garantice las cuatro libertades antes mencionadas, y desde ese momento el proyecto se podrá considerar libre. En el caso de que, por cualquier razón, no se posean todos los derechos sobre el mismo, habrá que recabar el consentimiento de todos los propietarios.

Dado que el mundo del Derecho es muy complicado, y que puede haber muchos matices, existen una gran cantidad de licencias redactadas. Las hay genéricas y específicas para código fuente, documentación, así como para otros tipos de contenidos como expresiones gráficas o sonidos. Los conceptos de lo que podemos hacer con un contenido libre son fáciles de entender, pero averiguar si una licencia se puede considerar libre o no, puede llegar a ser bastante engorroso. Por esta razón, el proyecto Debian [7] redactó una guia básica de las condiciones que debe cumplir una licencia

para considerarse libre, las directrices de software libre de Debian (Debian Free Software Guidelines, DFSG). Los abogados especializados de este proyecto analizan las diferentes licencias para ver si cumplen las DFSG. Sólo aquellos programas o contenidos que cumplen las directrices son incorporados a la parte principal del sistema operativo Debian GNU/Linux, y la comunidad del software libre adopta este mismo criterio para determinar qué licencias son libres.

Para el software en sí mismo, por lo general se suelen usar las del proyecto GNU [8], no en vano son los primeros promotores del software libre, y todo el mundo conoce sus licencias. Existen dos licencias para código, la GPL y la LGPL.

La licencia GPL (General Public License) es la original y la que más protege la obra. Otorga las cuatro libertades del software libre, pero con una condición importante: las modificaciones o mejoras que se hagan al software podrán ser distribuidas sólo en los mismos términos en los que se ha recibido el software original, esto es, licenciando las modificaciones a su vez con la licencia GPL.

La licencia GPL está muy bien para programas, pero cuando esta licencia se usaba en bibliotecas de programación, la obligación de liberar todo aquello que se enlace con código GPL podía representar un problema. Por ejemplo, podría llegar a ser ilegal crear cualquier tipo de software no libre para GNU/Linux. Por esta razón se creo la licencia LGPL (Lesser General Public License), que es igual a la GPL pero permite que los trabajos derivados de un código bajo licencia LGPL sean licenciados con otras licencias.

Para la documentación se ha estado usando durante bastante tiempo la misma licencia que para el código, pero en GNU pensaron que podía ser conveniente crear una licencia especial para documentación. Como resultado aparece la FDL (Free Documentation License), y aunque resulte paradójico viniendo de GNU, esta licencia tiene características que pueden violar las directrices de Debian (DFSG). Por lo tanto en Debian siguen recomendando [9] utilizar la misma licencia para el código y la documentación, apuntando además que esto tiene la ventaja de permitir copiar partes de código fuente a la documentación y viceversa.

Las licencias de GNU están traducidas en varios idiomas con objeto de que se comprendan. Sin embargo, en las traducciones, no está garantizado que su valor legal sea exactamente el mismo que el de la original en inglés, por lo que se recomienda usar la original.

Para aplicar las licencias al código, documentación o cualquier otro contenido del proyecto, basta con seguir las instrucciones que figuran en el apéndice de las licencias y con mayor detalle en las páginas web de las mismas.

Es importante señalar, que el hecho de liberar un PFC, no implica perder la posibilidad de explotarlo comercialmente. Aunque se está dando la oportunidad de usarlo a cualquiera, los autores serán con toda seguridad los más indicados para realizar la implantación, el mantenimiento, la ampliación, o dar soporte del sistema desarrollado [10,11].

## IV. Integración del proyecto en la comunidad

Un PFC libre alcanza una mayor utilidad para la Sociedad, y también probablemente para su autor, cuando es difundido. La forma más simple de publicarlo es simplemente ponerlo en una página web. Hoy en día existen estupendas herramientas de recuperación de información (buscadores) que probablemente permitirán localizar el proyecto a la gente interesada en él.

Si se quiere, se puede dar un nivel mayor de integración del PFC en el mundo del software libre abriendo el desarrollo a la comunidad[12]. Aunque requiere un esfuerzo mayor, esto puede dar continuidad al proyecto y por tanto más importancia y utilidad. No olvidemos que un programa informático nunca se termina, siempre es mejorable, o requiere mantenimiento para adaptarse a los continuos cambios del entorno. Además, puede ser muy gratificante ver cómo otros lo usan, e incluso colaboran en su desarrollo, y cómo lo que uno ha comenzado puede llegar a ser utilizado en todo el mundo.

En el modelo habitual de desarrollo de software libre suele haber colaboraciones de varios desarrolladores, publicación de versiones frecuentemente, informes de fallos o sugerencias de usuarios, y decisiones abiertas sobre la evolución del proyecto.

Para facilitar el desarrollo se utilizan diversas herramientas, y es casi imprescindible tener una página web del proyecto. En ésta se presentará el proyecto, indicando en qué consiste, el estado en que está el desarrollo y las ideas o planes de desarrollo futuro. Por supuesto, se incluirá en la página web un apartado para descargar tanto el programa como su código fuente, así como un apartado para la documentación, ya sea en línea o descargándola, para consultarla localmente o imprimirla. También se suele incluir un listado de tareas por hacer, instrucciones sobre cómo colaborar, agradecimientos de las colaboraciones recibidas, e información sobre quienes son los desarrolladores principales. Una herramienta interesante, a la que normalmente se podrá acceder desde la web, es el sistema de seguimiento de errores (“bug tracking system”). Éste permite: a los usuarios, informar de los problemas que encuentren en el software; y a los desarrolladores, llevar el control de los problemas que se van arreglando. Finalmente, se suele incluir un apartado para desarrolladores con todos los documentos del desarrollo (diseño, ideas, normas o estilos a seguir...), así como indicaciones sobre la forma de acceder a todos los recursos de desarrolladores. Normalmente, entre estos recursos se cuenta con un almacen de código fuente con sistema de control de versiones (CVS, Subversion). Se suele disponer también de diversas herramientas de comunicación, como las listas de correo, tanto para desarrollo como para dar soporte a los usuarios; o como canales de IRC que también pueden ser usados para ambos fines.

La tarea de poner los servicios comentados en marcha podría ser compleja, pero se simplifica mucho, usando alguno de los portales web que integran todas estas herramientas, y las ponen a disposición de los proyectos de software libre gratuitamente. Hay bastantes, entre ellos los

más conocidos son: SourceForge[13], Savanah[14] del proyecto GNU, y Software-libre.org[15] en español.

Hasta ahora hemos considerado la liberación de un proyecto nuevo, pero hay otras dos opciones, que incluso pueden ser más enriquecedoras. La primera es que el PFC se ligue de alguna forma a un SL ya existente. Un ejemplo de esto podría ser el desarrollo de un interfaz gráfico para un reproductor de música con interfaz de línea de órdenes. La segunda es que el PFC se integre completamente en un proyecto libre existente. Se puede considerar que el PFC sea la incorporación de una funcionalidad concreta, o el desarrollo de un cierto módulo necesario en el proyecto libre. En este caso, será muy importante concretar claramente los objetivos del PFC, diferenciándolos de los del proyecto libre en que se integra. También habrá que tener en cuenta que, a veces, para contribuir aportando código a un proyecto libre se nos pide que cedamos los derechos a una determinada organización, ésta garantizará a cambio que se mantiene como software libre. Será conveniente comprobar que se poseen todos los derechos sobre el código a integrar y las condiciones de la cesión.

La realización de un PFC de este tipo puede proporcionar al alumno que lo desarrolla una experiencia de trabajo real muy valiosa, compartiendo información y código con otros, y utilizando las diversas herramientas de trabajo en grupo que antes hemos comentado. Además, le proporcionará un conocimiento de primera línea sobre la comunidad del software libre y todo este fenómeno que todavía es una novedad en el ámbito empresarial. Lo que puede resultar beneficioso de cara a su carrera profesional porque cada vez hay más empresas trabajando en SL.

Otra ventaja es la reutilización. Al igual que nuestro código podrá ser reutilizado al ser libre, nosotros podremos utilizar otros códigos libres, consiguiendo realizar mejores proyectos con el mismo esfuerzo.

Es necesario que quede muy claro, haciéndolo constar en la memoria y en la presentación, qué es lo que aporta el PFC que se está realizando y qué partes son externas. A pesar de ello, la reutilización sigue siendo ventajosa, no sólo porque el PFC pueda aparentar tener mayor envergadura, sino principalmente porque se estará demostrando capacidad para trabajar en un entorno real en el que ya existe trabajo hecho por otras personas, y se demostrará capacidad para realizar una tarea más importante al conseguir hacer más cosas en el PFC.

La elección del tema del PFC la realiza el alumno, bien a propuesta propia, o bien eligiendo entre los ofertados por los profesores. En ambas opciones ya han aparecido iniciativas con software libre. Hay demanda por parte de los alumnos, que podemos comprobar leyendo algunos foros en internet, y hay profesores, como los del Grupo de Sistemas y Comunicaciones del Departamento de Informática de la Universidad Rey Juan Carlos, que ofertan temas de software libre.

Una última consideración sobre la elección de tema es que en el mundo del software libre se suele evitar “reinventar la rueda”. Es decir, no se suele iniciar un proyecto de software libre repetido, salvo que se crea que lo que ya existe no puede ser reconducido y el sistema nuevo puede aportar ventajas considerables.

## V. Conclusiones

El proyecto fin de carrera es una magnífica ocasión para el alumno de afrontar un trabajo real muy cercano a su entorno profesional. La experiencia del SL en este sentido puede ser aprovechada como vía que propicia un trabajo completo en varias líneas:

- Análisis de algo ya existente
- Definición clara de líneas de mejora
- Diseño de esta mejora dentro de un marco real.
- Verificación del resultado en un medio en el que incluso podrían participar agentes exteriores.
- Enriquecerse con una experiencia solidaria en un campo aparentemente “sin alma” por lo sofisticado y tecnificado.

## Referencias


[1] Real Decreto 1459/1990, de 26 de octubre, por el que se establece el titulo universitario oficial de Ingeniero en Informática y las directrices generales propias de los planes de estudios conducentes a la obtención de aquel.

[2] Torralba Martinez, J.M.(1992) Proyectos de Ingeniería Informática. En *Actas del VIII Congreso Nacional de Ingeniería de Proyectos*., páginas 17-30.

[3] Mantellán Olivera V. ¿Qué tiene que estudiar un informático? *TodoLinux*, nº 23, páginas 12-13. (También en [16])

[4] René Mérou. Argumentos en favor del software libre en las aulas, http://bulma.net/body.phtml?nIdNoticia=1831

[5] Grupo de Usuarios de Linux de la Universidad Carlos III de Madrid. Cómo hacer el Proyecto Fin de Carrera con software libre, http://gul.uc3m.es/gul/cursos/doc/PFCconSftwareLibre.pdf

[6] LEY 22/1987, de 11 de noviembre, de Propiedad Intelectual. BOE 275/1987 de 17 de noviembre.

[7] El proyecto Debian, http://www.debian.org

[8] El proyecto GNU, http://www.gnu.org

[9] DFSG and Software License FAQ, http://people.debian.org/~bap/dfsg-faq.html

[10] Jesús M. González Barahona. El software como servicio. O cómo producir programas libres y no morir en el intento. TodoLinux, nº 25, páginas 12-13. (También en [16])

[11] Juantomás García, Alfredo Romeo, Cristóbal Prieto. La pastilla roja. EditLin, 2003 ISBN:84-932888-5-3

[12] Jesús M. González Barahona. ¿Y cómo hago para que mi código sea libre? TodoLinux, nº 30, páginas 12-13. (También en [16])

[13] SourceForge, https://sourceforge.net

[14] Savannah, http://savannah.gnu.org

[15] #software-libre, http://software-libre.org/

[16] Vicente Mantellán Olivera et al. eds. Sobre software libre: Compilación de ensayos sobre software libre. Dijusa. 2004 ISBN: 84-9772-402-X

# SMB Web Client, Abriendo Windows a la WWW

Víctor Manuel Varela Rodríguez <vmvarela@nivel0.net>

***Resumen – SMB Web Client* es un interfaz web para acceder a redes Windows. Gracias al modelo de desarrollo abierto este script se ha convertido en menos de un año en una herramienta popular e imprescindible para cientos de administradores de sistemas en todo el mundo.**

## I. ¿Qué es?

Si te diriges a la web de SAMBA (http://www.samba.org) podrás leer su lema: "Opening Windows to a Wider World" (Abriendo Windows a un Mundo más Amplio). Sin ánimo de comparar mi sencillo script con esta maravilla del software libre, el lema de SMB Web Client podría ser "Opening Windows to the World Wide Web" (Abriendo Windows a la WWW).

*SMB Web Client* es un sencillo script escrito en PHP que permite utilizar servicios de ficheros e impresoras en una red local SMB/CIFS (Windows).

El objetivo principal era dotarme de una herramienta fácil de instalar y que, en ciertas configuraciones, me evitara tener que implantar una VPN para acceder a recursos remotos.

## II. Administradores de Sistemas: ¿Cómo se instala?

Para instalar SMB Web Client se necesitan 3 cosas:

1. smbwebclient.php, script que podemos descargar de la página http://www.nivel0.net/SmbWebClient.
2. smbclient, comando del paquete SAMBA (no es necesario tener instalado el servidor)
3. servidor web con PHP 4.1+ (se recomienda Apache 1.3)

Copia el archivo smbwebclient.php a una carpeta dentro de tu servidor web, edítalo según tengas configurado tu sistema y ya puede llamarlo desde el navegador.

### *II-A. Variables de configuración*

*cfgSambaRoot:* Puedes cambiar esta variable para limitar el acceso a un solo dominio (o grupo de trabajo), a un servidor, a un recurso compartido o a una carpeta concreta. Es decir, todos los recursos a los que accedas estarán por debajo. Por ejemplo, para que smbwebclient.php solo pueda acceder a la carpeta "Bilbo" en el recurso compartido "Bolson" dentro del servidor "HOBBIT" y del dominio (¡como no!) "TIERRAMEDIA" debes poner:

```
var $cfgSambaRoot = 'TIERRAMEDIA/HOBBIT/Bolson/Bilbo';
```

*cfgAnonymous:* Los valores posibles son "on" y "off". Para permitir el acceso sin contraseña:

```
var $cfgAnonymous = 'on';
```

Víctor M. Varela trabaja como programador en PHP y administrador de sistemas en la empresa Integración de Metodologías y Sistemas (http://www.netims.com)

ISBN 84-88783-71-X©2004

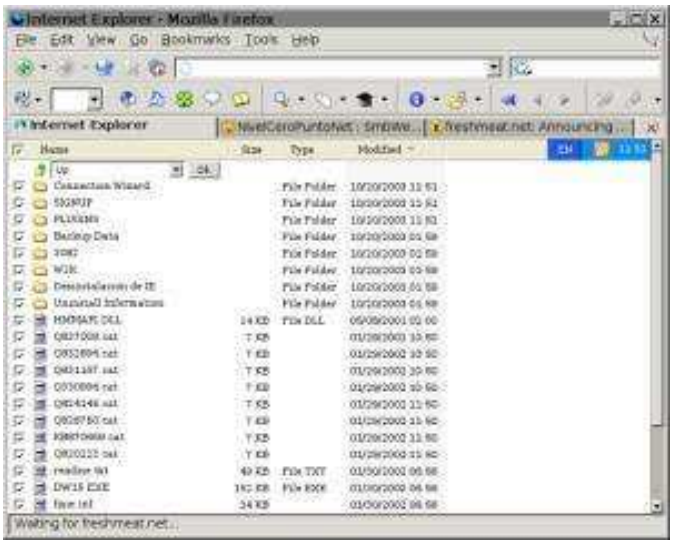


*Figura 1. Vista de carpeta*

*cfgCachePath:* Si esta variable es distinta de "" indica la ruta dentro del servidor donde se guardan los archivos descargados (a modo de caché) para acelerar la descarga. Si es "" se inhabilita la caché.

*cfgDefaultLanguage:* Idioma por defecto (el script normalmente es capaz de averiguarlo del navegador). El español sería "es".

*cfgDefaultServer:* Se trata del servidor que se va a utilizar para "ojear" la red. Por defecto vale "localhost", asumiendo que tienes instalado un servidor SAMBA en la misma máquina donde se ejecuta el script.

*cfgSmbClient:* Ruta del comando `smbclient`.

*cfgAuthMode:* Modo de autentificación. Puede ser `SMB_AUTH_ARG` (por defecto) o `SMB_AUTH_ENV`. El primero funciona en mayor número de sistemas por defecto pero es mucho más inseguro, ya que incluye la contraseña en la llamada a smbclient. El segundo método es más seguro (utiliza la variable de entorno USER) pero da problemas en algunos sistemas.

*cfgBaseUrl:* Por defecto es "smbwebclient.php" e indica la ruta de acceso web al script.

Si está activado el módulo mod_rewrite en Apache (ver httpd.conf) tienes la opción de crear un archivo ".htaccess" en el mismo directorio del script con el siguiente contenido:

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^(.*/[^\./]*[^/])$ $1/
RewriteRule ^(.*)$ smbwebclient.php?path=$1 [QSA,L]
```

De esta forma, cambiando el valor de cfgBaseUrl a 'http://servidor-web/ruta-carpeta-script/' podrás acceder a los recursos Windows con URLs "bonitas" de la forma DOMINIO / SERVIDOR / RECURSO / RUTA.

*cfgHideDotFiles:* No muestra los archivos “ocultos” de Unix (los que empiezan con “.”)

*II-B. Seguridad*

Para mejorar la seguridad recomiendo:

1. Utilizar el protocolo HTTPS
2. $cfgAnonymous = 'off'
3. $cfgCachePath = ''
4. $cfgAuthMode = 'SMB_AUTH_ENV'

## III. Programadores: ¿Cómo está hecho?

SMB Web Client esta escrito en PHP y realiza llamadas al comando `smbclient`, parseando su salida.

*III-A. Todo en un solo fuente*

La idea de incluir todo en un script trata de facilitar la actualización y la instalación del software. Para lograrlo, el script tenía que hacerse referencia a sí mismo en las etiquetas IMG. El problema era incluir los archivos de imagen dentro del mismo script, y para ello utilicé las funciones base64_encode y base64_decode de PHP. Hasta aquí bien, pero, ¿como editarlos?. Ok, lo que hice fué establecer dos modos de funcionamiento: (1, para distribución) utilizaba los archivos internos codificados en base64, y (2, para desarrollo) hacía referencia a los archivos externos. Para actualizar los archivos internos, que se encuentran en el array `inlineFiles`, programé un script de mantenimiento makefile.php, que enviaré a todo el que me lo solicite por email.

*III-B. Orientación a Objetos*

Desde versiones muy tempranas me solicitaron que empleara OOP para separar la interfaz web de las llamadas a `smbclient`. Ahora tenemos dos clases *samba* y *smbwebclient*, que hereda de *samba*.

*III-C. Smbclient*

Para obtener la información de smbclient se utiliza el operador `comando`. El parseado está realizado con la función preg_match (por su eficiencia), comparando cada línea con un array de expresiones regulares que facilita el mantenimiento.

*III-D. Syslog*

En versiones recientes estoy utilizando la función syslog para producir mensajes de depuración. La ventaja importante es que se integra con el resto de mensajes de “log” del sistema operativo.

*III-E. Traducción*

SMB Web Client está disponible en 36 idiomas a fecha de hoy. Este hito se ha logrado instalando un script de traducción en la misma página web de descarga http://www.nivel0.net/SmbWebClientTranslation. Uno se sorprende de lo mucho que puede colaborar la gente si se lo pones fácil.

## V. Software Libre (= Agradecimientos)

La rápida difusión de SMB Web Client se ha debido, en gran medida, a su publicación en Freshmeat, PHPClasses y SourceForge.

Si SMB Web Client ha podido ha sido gracias a la gran cantidad de usuarios que han reportado errores, enviado parches y traducciones y aportado ideas y sugerencias. Desde aquí mi más profundo agradecimiento a todos ellos.

# Un proyecto de gestión integral de tiendas usando código abierto

José Manuel González Vida

***Resumen*— La contribución que presentamos aborda un proyecto en desarrollo: la creación de un software de gestión integral de comercios que sea desarrollada íntegramente con software libre y que pueda ser gestionada en cualquier plataforma. La idea que subyace por debajo del proyecto es la de que la interfaz entre el usuario y el programa sea un navegador web. Por este motivo la aplicación está construida tomando como punto de partida una base de datos de MySQL sobre la que se construyen los distintos módulos usando diversas clases de PHP.**

## I. Introducción

El proyecto que presentamos surgió hace varios meses debido a la necesidad de tener interconectadas en tiempo real a dos zapaterías de Priego de Córdoba (ver [1]). Tras estudiar las distintas posibilidades que el mercado nos ofrecía se decidió interconectarlas usando una red WLAN basada en el estándar IEEE 802.11g. Es en este momento cuando comenzó a tomar cuerpo el software que se presenta en esta contribución. Los requisitos que se exigieron a este software fueron los siguientes: rapidez, seguridad, fiabilidad y fácil instalación y mantenimiento.

Para satisfacer estos requisitos el sistema que se propuso se basa en que la interfaz entre el usuario y el programa fuese un navegador web. Esta interfaz facilita en gran manera el mantenimiento del programa, ya que se pueden tener tantas terminales como se quieran y sólo es necesario actualizar el servidor central, pero añade una serie de dificultades debido a las propias limitaciones del contenido e interacción que el usuario puede hacer con un programa cuya interfaz se tiene a través de un navegador. Con la idea de que el sistema fuese desarrollado usando software libre así como dotarlo de la mayor estabilidad y robustez posible se ha desarrollado en un entorno linux en el que se utiliza un servidor con Apache 2 que sirve las páginas de la aplicación que han sido implementadas en PHP 4.3 (ver [2]) para extraer información de una base de datos creada con MySQL 4 (ver [3]) y complementado con diversos módulos que bien corren bajo PHP, o bien son bibliotecas externas.

Dpto. de Matemática Aplicada de la Universidad de Málaga (jgv@uma.es).

ISBN 84-88783-71-X©2004

Aunque para el desarrollo de esta aplicación se podrían haber empleado tecnologías más novedosas, la razón del uso de PHP junto con MySQL se debe a la madurez que ya tienen en el campo del software libre estas tecnologías, así como su capacidad de integración en los navegadores web.

Como más adelante se detalla, con el fin de proporcionar un aspecto más amigable así como de incrementar la eficacia de este programa ha sido necesario programar algunos módulos usando HTML 4.01 y CSS2, por lo que estos son los estándares que debe soportar el navegador para poder usar esta aplicación.

Finalmente el software que se está desarrollando se está convirtiendo en un programa de alto nivel para la gestión integral de tiendas que puede ser usado en múltiples plataformas y realizado íntegramente con software libre.

## II. El núcleo de la aplicación.

### *II-A. Entrada de nuevas referencias.*

El programa de introducción de nuevos artículos en la base de datos es seguramente la herramienta que incluye más funcionalidades de la aplicación. Cada nueva referencia que se introduce lleva asociada una imagen que es convertida a formato Jpeg y redimensionada automáticamente a varios tamaños predeterminados por la aplicación. Este proceso se lleva a cabo gracias a la introducción en PHP de la biblioteca GD v2.0 (véase [4]) que ha sido recompilada con el paquete jpeg-6b.

Una vez introducidos los datos de la nueva referencia se le asigna de modo automático un código de 13 dígitos numéricos que servirá para generar su correspondiente código de barras posteriormente.

Cabe destacar en este apartado que, aunque el programa ha sido pensado para un comercio de zapatos es trivialmente adaptable a cualquier comercio cuyos artículos tengan tallas e incluso, con pequeños cambios a un comercio en el que las referencias no tengan tallaje.

### *II-B. Generación de etiquetas.*

Otra de las funcionalidades del módulo de entrada de datos es la generación automática de etiquetas. Cada


Fig. 1. Módulo de introducción de datos.

etiqueta hace referencia a un producto único (no sólo a la referencia, sino también en nuestro caso a la talla del artículo) y contiene una imagen y un código de barras asociado al mismo. El resultado final de este módulo es la obtención automática de un documento en formato PDF con todas las etiquetas de los artículos introducidos y listo para imprimir.

Para ello se ha empleado una función que añade códigos de barras a documentos PDF programada por Valerio Granato (ver [5]). Esta función emplea la clase Cpdf de Wayne Munro (ver [6]) y la salida que devuelve el ejecutable GNU-barcode para insertar la imagen del código de barras deseado en la etiqueta. El resto de elementos que aparecen en la etiqueta (imagen, texto de distintas fuentes, etc.) se han introducido haciendo uso de los objetos de la clase Cpdf mencionada anteriormente y de la biblioteca GD ya mencionada antes. La unión de estos elementos para la generación de etiquetas consigue un resultado muy versátil, configurable fácilmente a cualquier necesidad y sobre todo muy cómodo.


Fig. 2. Ejemplo de etiqueta.

### *II-C. El módulo de TPV.*

Otro de los retos importantes al que nos enfrentamos en el diseño de esta aplicación fue el de la creación del módulo del terminal punto de venta. Este módulo ha de ser de muy fácil manejo, cómodo y rápido ya que el proceso de cobrar un artículo ha de ser muy eficaz. Además a todos estos requisitos hay que añadir que la interfaz entre el vendedor y el programa va a ser un navegador web (finalmente hemos usado Konqueror).


Fig. 3. El módulo Terminal Punto de Venta.

La programación de la interfaz de este módulo ha sido realizada empleando HTML 4.01 y CSS2 para evitar que haya múltiples marcos en la misma página. Se han usado scripts de Javascript 1.3 para ayudar al usuario cambiando el color de fondo de la casilla a rellenar en cada momento, etc. Tal vez la mayor dificultad en la implementación de este módulo ha sido el control de la impresión de los tickets simultáneamente al proceso de venta. La idea finalmente seguida ha sido la de instalar el servidor de impresión Cups tanto en el servidor como en los clientes habilitando la opción de compartir y descubrir las impresoras remotas. De este modo el ordenador cliente desde el que se está haciendo la venta se identifica frente al servidor con su dirección IP y por tanto el servidor conoce dónde tiene que enviar el ticket de la venta en curso.

## III. Trabajo Futuro

Esta aplicación está siendo usada por los comercios que se mencionan al principio de este artículo con mucho éxito debido a la velocidad con la que se obtienen las respuestas del programa ya que prácticamente no se distinguen frente a una aplicación que trabajase en el ordenador local.

Actualmente estamos trabajando en completar las funcionalidades de esta aplicación incorporando secciones para controlar las estadísticas de los productos más vendidos, generación de reposiciones automáticas entre tiendas, posibilidad de incorporación de un módulo de gestión-fidelización de clientes, adaptación del módulo TPV a una pantalla táctil, etc.

Uno de los módulos que se están poniendo en marcha actualmente en coordinación con la empresa T2V S.L. es la sincronización de los datos desde el servidor central de las tiendas con un servidor web de esta empresa para que simplemente activando una casilla en cualquier producto éste pueda ser visto en tiempo real a través de la tienda virtual que estas zapaterías tienen en internet.

Otra de las líneas de trabajo que estamos haciendo es la de intentar demostrar que es posible realizar una aplicación de fácil manejo, intuitiva y eficaz íntegramente con software libre y destinada al pequeño y mediano comercio.

## Referencias

[1] “González-vida, zapaterías,” http://www.gonzalez-vida.com.
[2] “Php, the hipertext preprocessor,” http://www.php.net.
[3] “Mysql,” http://www.mysql.com/developer.
[4] “Gd graphics library,” http://www.boutell.com/gd.
[5] Valerio Granato, “Pdf-barcode,” http://www.grana.to/pdfbarcode.
[6] Wayne Munro, “Cpdf class,” http://www.ros.co.nz/pdf.

# DamFlow: un software de visualización de Flujos Hidrodinámicos desarrollado con Python, C++ y VTK

Ana María Ferreiro Ferreiro[1], José A. García Rodríguez[2]

***Resumen*— En este artículo describimos el desarrollo, usando código abierto, de un software de postproceso multiplataforma, implementado para analizar resultados de simulaciones de flujos hidrodinámicos obtenidos en un cluster de PC's bajo entorno Linux.**

## I. Introducción

Hoy en día merced a la potencia y abaratamiento del hardware es posible abordar gran cantidad de problemas en diversas ramas de la ciencia mediante el uso de simulaciones llevadas a cabo con clusters de ordenadores. Producir estos resultados y almacenarlos no deja de tener una gran dificultad en muchos casos. Pero a esta dificultad se añade que, si bien puede resultar difícil la obtención de simulaciones que modelen problemas realistas, puede resultar no menos difícil el analizar los resultados obtenidos. En muchas ocasiones analizar es visualizar, pero no sólo obtener gráficas, sino permitir la interacción del usuario con el software de modo que le sea posible centrar su atención en los detalles de interés: crear animaciones en tiempo real, extraer series temporales de datos en los lugares deseados, etc. La idea de DamFlow (véase [1]) es proporcionar a alguien que no tiene por qué conocer cómo se ha realizado una simulación y que quizás pertenezca a otro área de la ciencia, herramientas que le permitan analizar e interpretar lo que está viendo. Por tanto es indudable la utilidad de este tipo de software que ahorra a los científicos gran cantidad de tiempo al interpretar los datos que obtienen y permite comunicarlos a otros científicos.

Existen en el mercado numerosas herramientas de postproceso (que muchas veces engloban también facilidades de preproceso y de cálculo). Indudablemente muchas de ellas son de una gran calidad, pero van acompañadas por un alto coste que no todos los investigadores pueden asumir. Que algo sea caro tampoco significa que cumpla todas las necesidades específicas que tengamos. Es por ello mejor poder disponer de software a medida realizado para los problemas concretos a tratar. En este punto es donde juega su papel el software libre.

El software que presentamos ha sido realizado en el seno del grupo de investigación EDANYA de la Universidad de Málaga. El trabajo de este grupo de investigación se centra en el estudio, modelado y análisis de flujos hidrodinámicos. Los investigadores de este grupo desarrollan modelos numéricos para simular el comportamiento de masas de fluidos: desde la simulación del intercambio de flujos a través del Estrecho de Gibraltar, hasta la simulación de catástrofes por inundaciones, etc. Al abordar simulaciones en geometrías realistas (en entornos 3D) surgió la necesidad de tener herramientas de análisis de estos resultados que fuesen flexibles y permitiesen tratar ficheros de resultados que pueden contener muchos gigas de información. Los resultados que se guardan de estas simulaciones suelen ser enormes series temporales de datos de altura del agua, velocidades, concentraciones de sustancia, y cualquier propiedad del fluido que pudiese ser necesario estudiar.

En algunas ocasiones estas simulaciones pueden corresponder a tiempos reales de meses o años (las mareas en el Estrecho de Gibraltar, por ejemplo). Esto genera enormes ficheros de datos.

Las simulaciones se llevan a cabo en un cluster de PC's cuyo sistema operativo es SuSE Linux. Los ordenadores se conectan mediante una red Gigabit y los nodos hijos carecen de Disco duro (diskless cluster), mientras que el maestro tiene discos UWSCSI 320. Este cluster ni siquiera se encuentra físicamente cerca de los científicos que lo usan, por tanto la visualización de los resultados ha de realizarse en remoto, ya que no podemos trasladar los datos a través de la red cada vez que queremos ver una simulación pues ello supondría mucho tiempo de espera.

Para realizar el postproceso de todos los datos generados en las simulaciones se decidió implementar un software modular basado en herramientas de código abierto y que fuese fácil de modificar para adaptarlo a

1 Universidad de Málaga (anafefe@anamat.cie.uma.es).
2 Universidad de Málaga (joseanto@anamat.cie.uma.es).

ISBN 84-88783-71-X©2004

los distintos tipos de problemas que se abordan en este grupo de investigación.

Tras un proceso de evaluación de distinto software de código abierto, se decidió implementar DamFlow usando como lenguaje base Python, como kernel de visualización VTK y como acelerador del proceso de visualización C++ (por medio del generador automático de wrappers ”Weave”(ver [2])).

## II. ¿Por qué Python?

Python (ver [3]) es un lenguaje de programación de tipo interpretado, parecido a Java. Tiene una máquina virtual (la PVM, Python Virtual Machine) y se puede ejecutar también en modo consola de manera parecida a Matlab. Lo que lo hace único es su productividad sin precedentes. Python es un lenguaje orientado a objetos de alto nivel para el que es posible encontrar envolturas (”wrappers”) para casi todos los lenguajes de programación. Esto hace posible por ejemplo, escribir un programa en Java usando Python (la envoltura de Java para Python se llama Jython). Además cuenta con una enorme cantidad de módulos para realizar todo tipo de tareas. Esto hace que esté presente en una gran cantidad de ambientes: se usa tanto para programación científica (NASA, ESA, CERN) como para por ejemplo la industria audiovisual (Maya, Light and Magic, Disney), está presente en la implementación de Google o RedHat y como lenguaje de script para realizar aplicaciones web, etc. También muchos paquetes de software comercial como por ejemplo el Abaqus lo han adoptado como lenguaje de scripts.

Python por sí mismo no es un lenguaje rápido para la programación científica, pero tiene módulos programados en C, C++ y Fortran que lo hacen casi tan rápido como C++ si se usa correctamente. Un ejemplo son las bibliotecas que podemos encontrar en el Scientific Python (ver [4]), que incluyen desde la posibilidad de usar BLAS y LAPACK desde python, como de dibujar gráficas, etc. Además es posible construir nuestros propios módulos en C o C++ para después usarlos desde Python usando generadores automáticos de ”wrappers”.

## III. ¿Por qué VTK?

Para llevar a cabo la implementación del programa de visualización hemos empleado como núcleo gráfico el Visualization Toolkit (VTK). El Visualization Toolkit (VTK) es una librería gratuita para gráficos 3D y visualización. Las clases están implementadas en C++ y tiene una excelente documentación que está generada mediante Doxygen. VTK además tiene wrappers para casi todos los lenguajes, como por ejemplo: Tcl, Python, Java, etc... Para una descripción detallada de VTK véase [5].

## IV. Descripción del programa

La interfaz de DamFlow se presenta como una ventana (véase Figura 1), compuesta por 3 frames: una barra de menú superior con submenús desplegables a través de los cuales es posible acceder a todas las funcionalidades; una barra lateral a la izquierda con las utilidades referentes al tipo de visualización en curso y una ventana de renderizado a la derecha (un objeto de tipo vtkRenderWindow) donde el usuario puede interactuar con la simulación que está viendo. En el panel izquierdo se encuentran los botones que permiten visualizar la animación. Es posible rotar los objetos del render, desplazarlos y hacer zoom. También es posible obtener gráficas de valores seleccionando un punto geográfico.

Como necesitábamos generar una interfaz amigable, hubo que decidirse por un lenguaje en el que fuese rápido generar código. Elegimos Python porque es un lenguaje con el que se consigue una productividad a la hora de programar sin precedentes. Además Python a su vez tiene ”wrappers”para manejar una gran cantidad de toolkits de ventanas: entre ellos Tcl, Qt, Wxwindows, etc. Finalmente para implementar las ventanas decidimos utilizar Tkinter (una envoltura para usar Tcl/Tk desde Python), que es el estándar que ofrece Python para construir GUIs y viene incluido con cualquier instalación de Python. Nos decidimos por Tkinter porque de esta manera cualquiera que tuviese Python instalado podría instalar DamFlow. Además ya existían ciertas componentes reutilizables programadas en Tkinter y VTK que nos resultaron de gran utilidad (véase [6]).

Los resultados que se generan en el cluster son simulaciones de flujos hidrodinámicos. Al realizar simulaciones con mallados 2D fue necesario paralelizar los programas de simulación, puesto que el coste computacional al usar mallados bidimensionales se incrementa considerablemente. Como consecuencia, las simulaciones se guardan en varios ficheros cada uno de los cuales corresponde al trozo de geometría donde uno de los nodos hijos ha realizado los cálculos en el trozo de geometría que se le ha asignado. Así pues, es necesario leer estos trozos sueltos y regenerar la geometría completa a la hora de realizar la visualización tridimensional.

## V. Uso de C++ para mejorar el rendimiento

El programa permite la visualización de los resultados temporales en animación directamente desde los ficheros que contienen las simulaciones. Esto conlleva el que sea necesario rellenar los valores de los actores de la

escena de forma rápida para poder conseguir sensación de una animación en tiempo real y no tener tiempos de espera entre frames demasiado altos. El problema es que Python no es un lenguaje rápido (incluso programando cuidadosamente los bucles excesivamente largos lo pueden ralentizar en exceso). Por ello, para cualquier tarea que el programa necesite realizar de forma rápida fue necesario incluir código C++. Así pues, la mayor parte del programa está desarrollado en Python, sin embargo las partes más sensibles, donde los que se necesita es rendimiento, fueron reescritas en C++.

Hay varias maneras de poder introducir código C++, fortran o C en programas escritos en Python. Quizá la más conocida sea el generador de interfaces SWIG (ver [7]). También es posible emplear Boost o la PythonC API. Sin embargo, como el propósito era realizar la implemetación del software lo más rápido posible optamos por emplear Weave, una parte del Scientific Python que permite introducir código C++ directamente en Python (código ”in line”) compilándolo de forma automática una sola vez, generando así una librería a la que se llama en tiempo de ejecución. El Weave incluye además un fichero llamado vtk_spec.py que hace posible pasar objetos de vtk de Python a C++ y a la inversa, refiriéndose a sus posiciones de memoria.

Con lo cual lo que se hace en el curso de una sesión del programa es rellenar actores de VTK en C++ y devolverlos ya rellenos de nuevo al programa Python que se encarga de gestionarlos, de introducirlos en la escena y de permitir que los usuarios actúen sobre ellos de forma interactiva. Con esto conseguimos un gran incremento en el rendimiento de la aplicación, lo que hace posible visualizar geometrías complejas.

## VI. Ejemplos de aplicación del software

En las capturas que se presentan en esta sección se puede ver el aspecto que tiene la interfaz gráfica.

Una vez que se ha elegido la simulación a visualizar basta seleccionar cualquiera de los ficheros correspondientes del directorio en el que se hayan guardado los ”trozos”de la simulación paralela. El programa busca cuales son los otros trozos que corresponden a la simulación y de manera automática muestra toda la geometría ya reconstruida.

En el panel que se despliega a la izquierda correspondien-te a la visualización en curso hay numerosas opciones. Es posible visualizar los mallados que conforman los objetos que se están visualizando.

También disponemos de paneles que nos permiten cambiar propiedades de la escena: color del fondo, añadir una leyenda que explique aquello que se está viendo (con fuentes antialiased), cambiar las luces que iluminan la escena y añadir más luces si es necesario.

Es posible visualizar las propiedades escalares del fluido, como salinidad, temperatura, etc. mediante escalas de colores que el usuario puede modificar interactivamente (e incluso si lo desea, puede crear sus propias tablas de colores).

Podemos hacer secciones de cualquier zona de los objetos que se encuentran en el render (ver Figura 3).

Es posible guardar animaciones de las simulaciones que se están visualizando, y también capturas en diversos formatos gráficos: png, jpeg, tiff, gif, etc. También es posible hacer capturas en formatos gráficos vectoriales (eps, ps,...).

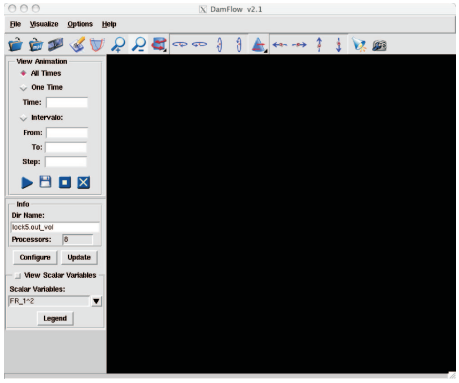



Fig. 1. Aspecto de la interfaz de DamFlow.

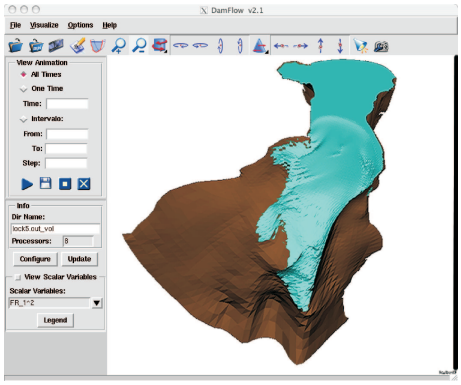
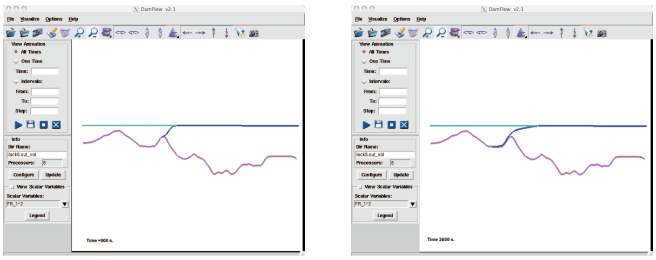



Fig. 2. Experimento en el Estrecho de Gibraltar.

## VII. Trabajo futuro

Este software, hasta ahora sólo aporta capacidades de postproceso: de análisis de las simulaciones realizadas. Está proyectado realizar una plataforma más completa que permita además la visualización de datos de tipo GIS con sus correspondientes texturas. También resultaría de interés integrar facilidades de mallado (preproceso) que



(a) Tiempo 900 seg. (b) Tiempo 3600 seg.

Fig. 3. Ejemplo de una sección unidimensional de una simulación

permitiesen a los investigadores automatizar, en la medida de lo posible, la tarea de realizar simulaciones en geometrías realistas. Otra tarea pendiente es la introducción de algunas facilidades de CAD que permitan modificar las geometrías proporcionadas en ficheros de tipo GIS, por ejemplo, a fin de poder introducir estructuras como puertos, puentes, etc. cuya simulación pueda resultar de interés.

Esto haría posible el uso de las componentes de software hidráulico realizadas en el seno de este grupo de investigación a personas de otras disciplinas como Biólogos, Físicos, Oceanógrafos,... que podrían encontrar en este tipo de herramientas una ayuda para sus investigaciones.

Dado que el software de simulación para el que está diseñado la aplicación puede ejecutarse en paralelo en entorno cluster, también resultaría de interés aprovechar el hecho de que la aplicación corra en un cluster: podrían paralelizarse las tareas más pesadas, con lo que obtendríamos una visualización en tiempo real incluso trabajando con mallados o geometrías mucho más complejas.

## Referencias


[1] “Damflow: Dynamics advanced modelling system for environmental and hydraulic water flows,” http://www.damflow.org.
[2] “Performance python with weave,” http://www.scipy.org/documentation/weave/weaveperformance.html.
[3] “Python,” http://www.python.org.
[4] Enthought, “Scientific python,” http://www.scipy.org.
[5] Kitware, “Vtk (the visualization toolkit),” http://www.vtk.org.
[6] Prabhu Ramachandran, “Mayavi,” http://mayavi.sourceforge.net.
[7] “Swig,” http://www.swig.org.

# Taller de reutilización de PC's mediante PXES

Ignacio Montoya García, José Pardo Barceló

***Resumen.*** **– Mediante este taller se pretende dar una visión general sobre las posibilidades de reutilización de ordenadores obsoletos para construir redes de thin-clients. Para ello se hará uso de la herramienta PXES. Todo el software utilizado en este taller será software libre.**

## I. Introducción

La reutilización de ordenadores denominados como obsoletos mediante la creación de redes de clientes ligeros (thin-clients), además de beneficios indirectos de tipo social y económicos puede proporcionar una serie de beneficios directos tanto a los usuarios de dichas redes como al administrador de la misma.

En los últimos años se estan desarrollando de forma muy activa varios proyectos de software libre para facilitar la creación de dichas redes, proyectos tales como el proyecto LTSP [1], NETSTATION [2] ó PXES [3].

Este taller pretende dar una visión general sobre la necesidad y las ventajas de reutilizar ordenadores considerados como obsoletos, una introducción teórica básica al funcionamiento de una red de clientes ligeros así como el montaje de forma práctica de una red de este tipo mediante la herramienta de software libre PXES.

### *I-A Organización del taller*

Los participantes del taller se dividiran en grupos de un máximo de tres personas. A cada grupo se le asignarán dos ordenadores, uno de los cuales actuará de cliente y otro de servidor. El servidor tendrá acceso a internet. Durante el desarrollo del taller se conectará directamente el cliente al servidor mediante un cable cruzado.

## II. Requisitos previos

### *II-A Requisitos de Hardware*

Para una descripción detallada visitar la página de pxes [4]

Cliente:

1. Procesador: Arquitectura x86
2. RAM: mínimo 16Mb (recomendados 32Mb)
3. disquetera 3"5
4. Tarjeta de red
5. Monitor
6. Teclado
7. Ratón

Servidor:

1. Procesador: Arquitectura x86 (i686 recomendado)
2. RAM: mínimo 128 Mb (recomendados 256Mb)
3. Tarjeta de red
4. Tarjeta de video

Red:

1. Conexión en red entre el cliente y el servidor de cada grupo (mediante cable cruzado)

### *II-B Requisitos de Software*

Cliente:

1. Capacidad de arranque desde disquete (acceso a la BIOS en caso necesario a fin de modificar el arranque)

Servidor:

1. Sistema operativo gnu/linux
2. Entorno de escritorio gnome (con gdm)
3. Acceso al sistema como root.
4. Acceso a internet.

## III. Reutilización de PC's

Según Gartner Dataquest en Abril de 2.002 se fabricó el ordenador personal 1.000 millones desde que comenzó su producción. La misma consultora estima que el ordenador personal 2.000 millones se fabricará en algún momento del año 2.007. Esto implica que en el año 2007 se habrá construido un ordenador por cada 3 habitantes del planeta. Es decir, uno de cada tres habitantes del planeta según esto y dando por hecho un reparto equitativo de los recursos podria tener un ordenador personal para el año 2007. Esto, evidentemente, es una utopía. El mal reparto de los recursos y la infrautilización de los mismos nos conduce a una situación en la cual más del 90% de los habitantes del planeta no tienen acceso a un ordenador personal.

Ignacio Montoya Garcia pertenece al grupo “Círculos de Innovación y Tecnología” de la Universidad de Cádiz.
ignacio.montoya@uca.es
Jose Pardo Barceló pertenece al grupo “Círculos de Innovación y Tecnología” de la Universidad de Cádiz.
jose@inicianet.com

ISBN 84-88783-71-X

*III-A La brecha digital*

Un informe del Ministerio de Ciencia y Tecnología (MCyT) señala que las diferencias entre los países que tienen acceso real a las Tecnologías de la Información y las Comunicaciones (TIC) y los que no lo tienen, así como dentro de los propios países, siguen creciendo, por lo que la denominada "brecha digital" aumenta en lugar de reducirse.

"Aunque todos los países, incluso los más pobres, han incrementado su utilización de las TIC, los países desarrollados han avanzado de forma exponencial, de manera que las diferencias siguen aumentando", recoge el libro "La Sociedad de la Información en el siglo XXI: un requisito para el des@rrollo", impulsado por el MCyT.

Además, el informe recoge que este problema se mantiene dentro de cada país, donde "viene sucediendo algo parecido por lo que la brecha interna crece, también, de igual forma". Este es el caso entre núcleos urbanos y rurales, o entre clases sociales más o menos adineradas.


*figura 1: La brecha digital en el mundo*

El problema radica en que la "'brecha digital" implica que los pueblos que no tienen acceso a la información pierden una "oportunidad única de colmar sus necesidades básicas de desarrollo", y aquellos grupos excluidos que se encuentran en países desarrollados "también pierden opciones de progresar en todos los ámbitos", según añade el informe.

*III-B Infrautilización de PC's*

Según un estudio de la Carnegie Mellon University del año 1997 (*Disposition and End-of-Life Options for Personal Computers*, Carnegie Mellon University,1997), entre los años 1985 y 2005 quedarán obsoletos en Estados Unidos unos 325 millones de ordenadores personales (*Disposition and End-of-Life Options for Personal Computer*, Carnegie Mellon University,1997). De esta cantidad unos 55 millones acabarán en los vertederos, 143 millones se reciclarán y los restantes 127 millones serán reusados o almacenados.

Por otro lado, de acuerdo con RetroSystems Inc. [5], el tiempo de vida de la CPU de un PC es tanto menor cuanto más reciente sea su año de fabricación, estimando que esta vida se estabilizará alrededor de los dos años a partir del año 2.005.

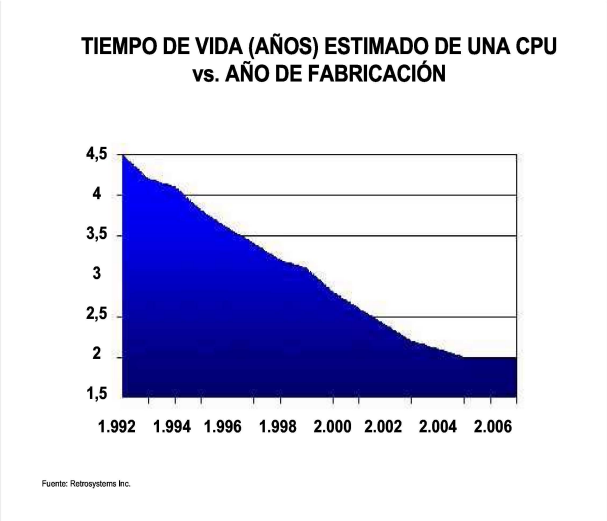



*figura 2 tiempo de vida estimado de una CPU Vs año de fabricación*

*III-B Reciclaje Vs Reutilización*

Se estima que para producir un *chip* de 2 gramos se hace precisa la utilización de 1.600 gramos de combustibles fósiles, 72 gramos de productos químicos, 32.000 gramos de agua y 700 gramos de gases. [WIL 02].

Para hacer un ordenador de sobremesa con un tubo de rayos catódicos de 17", se precisan al menos 240 kg de combustibles fósiles, 22 kg de productos químicos y 1.500 kg de agua. Los 240 kg de combustibles fósiles corresponden con 5.040 MJ en unidades estándar de energía

Se están desarrollando grandes esfuerzos para eliminar de forma respetuosa con el ambiente tan ingente cantidad de residuos. Pero, ¿sólo reciclar?

El reciclado de un ordenador sólo salva un 0,43% de la energía que se precisa para construirlo (debido a las materias primas). Esta impresionante diferencia nos indica claramente la necesidad de ampliar el período de utilización de los ordenadores (Computer and the Environment, p. 195) es decir, la necesidad de reutilizar en contraposición a reciclar.

## IV. Teoria General de Thin Clients

*IV-A Qué es una red de thin client*

Una red de thin clients es aquella que se constituye con equipos de bajo coste en el lado del cliente en los cuales tan solo se mostrará información y un equipo o varios en el lado del servidor que ejecutará y enviará por red el resultado de las operaciones realizadas por los usuarios. De esta forma se centralizan los recursos de la red haciendo mucho más cómoda su administración. Se podría resumir de forma sencilla diciendo que los thin client son terminales "tontas" desde las cuales se

ejecutan aplicaciones en el servidor de manera gráfica.

Los clientes podrán obtenerse por medio de reutilización informática o por compras de equipos nuevos (thin client).

Al iniciar un cliente éste envía una señal por la red en la que se encuentra identificándose con la MAC de su tarjeta de ethernet, tras lo cual permanece a la espera. El servidor al identificar la MAC envía los datos de asignación de red al equipo (nombre de host, ip, rutas, máscara de red, ...) y por medio de TFTP la imagen indicada en configuración. La imagen estará compuesta por todo lo necesario para el funcionamiento del cliente y se ejecutará en la RAM del mismo. Por último el cliente una vez iniciado accederá al login gráfico del servidor lo que le permitirá ejecutar cualquier aplicación que éste posea. Esto le permite al cliente funcionar sin disco duro.

*IV-B Ventajas e inconvenientes de una red de thin clients:*

Ventajas:

1. Reutilización de ordenadores obsoletos
2. Los clientes ligeros pueden ejecutar software cuyos requisitos mínimos de hardware están muy por encima del hardware que posee dicho cliente ligero.
3. Los clientes ligeros comparten un banco de memoria RAM común en el servidor para ejecutar las aplicaciones remotas. Esto permite un importante ahorro de memoria RAM y además hace que las aplicaciones más utilizadas sean las que se ejecuten más rápidamente.
4. Los clientes ligeros pueden correr distintos sistemas operativos sin necesidad de realizar ningún cambio en ellos.
5. La vida útil de los clientes ligeros es enorme (teóricamente mucho mayor que la de un pc de escritorio convencional)
6. Al ejecutarse todas las aplicaciones en el servidor central, los posibles costes de propiedad para la red entera se reducen a los costes de propiedad de un ordenador.
7. Los costes de mantenimiento de la red se reducen prácticamente a los costes de mantenimiento de un ordenador (dando por hecho que los clientes ligeros apenas se estropean).
8. La administración de la red en conjunto, una vez configurada por primera vez, se reduce a la administración del servidor.
9. La administración de las copias de seguridad se simplifica.
10. La implentación de mecanismos de seguridad se reduce a proteger un equipo.
11. Los puntos de fallo en general de la red (virus, malas actualizaciones, ataques externos, etc) se reducen a un equipo.

Inconvenientes:

1. El funcionamiento de la red depende totalmente del correcto funcionamiento del servidor. Si cae el servidor, cae la red entera.

## V. Parte práctica

*V-A Configuración del cliente*

Métodos de arranque:

Existen variados métodos para arrancar los clientes ligeros.

1. Ethernet.
2. disquete.
3. Disco duro.
4. Usb.
5. Cdrom.

Las mayoría de tarjetas ethernet tienen un socket en el cual es posible insertar una pequeña memoria eprom. En dicha memoria se escribir,ia el código del cargador de arranque a partir del cual puede arrancar el cliente. Aunque este método se puede considerar como el método "puro" de arranque para clientes ligeros, implica el uso de una grabadora especial de memorias eprom, motivo por el cual se descarta para el taller.

El método utilizado en el taller será un disquete de arranque basado en etherboot creado en la web de rom-o-matic [5]. Una vez creado el disquete de arranque se llevará a cabo una prueba de funcionamiento para comprobar que reconoce la tarjeta de red del cliente.

*V-B Configuración del servidor*

Instalación y configuración de un servidor dhcp:

Los pasos a seguir para instalar el servidor dhcp serán los siguientes:

1. Se descargará el servidor de la red.
2. Se instalará dicho servidor. En el caso de estar usando un sistema linux basado en Debian (como Guadalinex o Ubuntu), bastará con usar la orden *apt-get install dhcp*. De todas formas se explicará el método de instalación para sistemas basados en paquetería rpm o bien para sistemas sin sistema alguno de paquetes (uso de makefile).

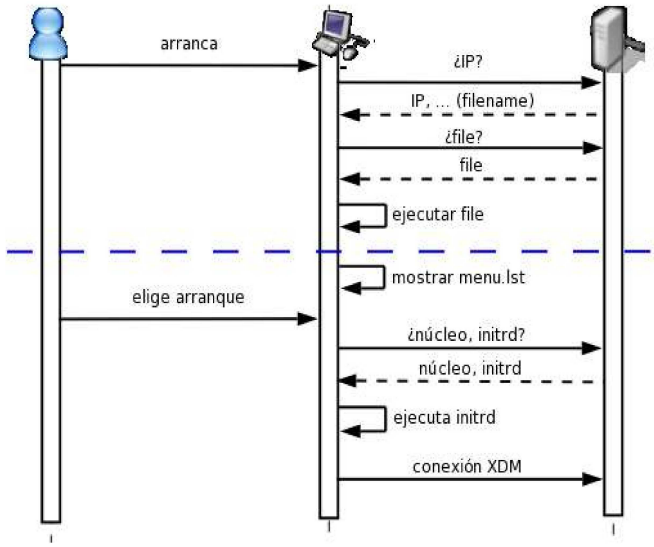



*figura 3: Proceso general de arranque de un thin client*

3. Se configurará el servidor modificando el archivo '/etc/dhcpd.conf'. A través de un fichero de configuración como ejemplo se verán las distintas opciones de configuración posibles, asi como las mas recomendables según cada caso particular.
4. Se procederá al reinicio del servicio dhcp.

Instalación y configuración de un servidor tftp:

Se hará lo siguiente:

1. Al igual que con el servidor dhcp, se descargará y se instalará el paquete correspondiente mediante *apt-get install tftpd-hpa.*
2. Durante la instalación se explicarán dos modos en que puede correr este servicio: demonio o xinetd.
3. Se crearán las estructuras necesarias para el servicio (directorio /tftpd) y se explicará la configuración del mismo.
4. Se procederá al reinicio del servicio.

Configuración de gdm:

1. Se configurará gdm para que admita sesiones remotas (en este caso desde el cliente ligero.

*V-C Instalación y configuración de PXES:*

Instalación de PXES:

Antes de empezar es importante comprobar si se tienen instaladas las librerias y programas:

- libgtk-perl
- glade-perl
- libgnome-perl
- mknbi

También se utilizarán un servidor DHCP y uno TFTP que deberán de estar instalados

Aunque existen paquetes preparados para algunas distribuciones es recomendable bajar los binarios pertenecientes a pxesconfig disponibles en: http://pxes.sourceforge.net/

Para instalar se hará tan solo:
# tar xzvf pxes-base-i586-X.X-X.tar.gz
# cd pxes-X.X
# make install
De la misma forma:
# tar xzvf pxesconfig-X.X-X.tar.gz
# cd pxesconfig-X.X
# perl Makefile.PL
# make
# make install

Generar imagen PXES:

La forma más sencilla de iniciar una terminal es usar las imágenes pregeneradas disponibles en la página de pxes.

El archivo /etc/fstab
Añadir en el archivo /etc/fstab
/tmp/pxes.initrd /tmp/pxes ext2 loop,noauto,user,owner 0 0

En la primera ejecución de PXESconfig será necesario la creación de una imagen .initrd

A lo largo del resto del taller se irá haciendo una descripción de las diferentes pantallas del configurador de pxes eligiendo en cada caso las opciones específicas al entorno donde se desarrolla el taller.

1. Selección del núcleo y de la imagen inicial:

- Splash screen

Pantalla gráfica que se mostrará durante el arranque. Es posible personalizarla.
*acción:* Se dejará el salvapantallas por defecto.

- Kernel

Kernel que iniciará el terminal. Se ha de seleccionar el más apropiado.
*acción:* Se dejará el kernel por defecto

- Initrd

Nombre del archivo de imagen inicial en disco RAM
initialize ram disk: Activado crea y guarda unos nuevos valores iniciales para el disco ram.
Read Save Configuration: Carga los valores del archivo . initrd creado con anterioridad.
*acción:*Se dejará la imagen por defecto

- Network bootable image

Introducir donde y con que nombre se guardará este tipo de imagen (ruta recomendada: /tftpboot/pxes/xxxx.nbi)
Enable Network bootble Image: Activar para crear una

imagen utilizable con el arranque desde disquete o red
*acción:* Se creará una imagen nueva activando la opción correspondiente.

- Iso Bootable image

Teclear la localización y el nombre que se la dará a la imagen para utilizar desde cd
Enable ISO 9660 bootable image generation: Activar para la creación de la imagen
*acción:* No hace falta esta imagen así que se no se seleccionará esta opción.

2. Configuración de dispositivos locales:

Seleccionar los dispositivos locales para que coincidan con los que poseen los clientes. Es posible la autodetección pero esto hará mayor la imagen ejecutada en la ram del cliente.

- Keyboard layout

Seleccionar el idioma del teclado local.
*acción:* Se buscará la opción 'es'

- Mouse device

Seleccionar el dispositivo y el protocolo a utilizar para el ratón local del terminal.
Es posible activar la rueda central o especificar el ratón para zurdos.
Indicar aceleración y sensibilidad
*acción:* Según el cliente habrá que seleccionar una de las posibles opciones. Las más comunes en cuanto al dispositivo serán '/dev/psaux' en caso de tener un ratón ps/2 y '/dev/ttyS0' en caso de tener un ratón serie. En cuanto al protocolo, se recomienda la opción 'auto'.

- Network Configuration

Permite Indicar la tarjeta de red utilizada en el equipo cliente.
*acción:* Se indicará el tipo de tarjeta usada por los clientes.

3. Configuración de dispositivos locales opcionales:

Dispositivos locales que controlará el cliente.

- Shared devices Configuration

Activando las casillas se permitirá el acceso a los dispositivos locales indicados.
*acción:* En principio no se permitirá el acceso a ningún dispositivo

- Sound System Configuration

Enable local sound support: Activo permite la reproducción de sonido en el cliente
Sound Card: Indicar la tarjeta de audio instalada en el cliente. Es posible la auto detección pero esto implicaría un mayor tamaño de la imagen.
*acción:* No se seleccionará esta opción

- Enable local printer support**:** Activa el soporte local para impresora.
Indicar el dispositivo y el puerto de escucha.
*acción:* Se dejará esta opción sin seleccionar.

- Samba

Activo permite el uso de Samba para compartir los dispositivos locales.
*acción:* No se seleccionará esta opción.

4. Selección de sesiones:

Selección de los tipos de conexiones que podrán realizarse desde el cliente.
En el caso de activar Local x-Window Session el sistema se iniciará en el cliente en modo local con una sesión x-Window y mostrará todas aquellas posibilidades de acceso que se hayan configurado en él. Todo el trabajo con sesiones se realizará en diferentes ventanas siendo posible tener varias abiertas al mismo tiempo.
En el caso de marcar sólo uno de los tipos de sesiones se accederá a la pantalla de identificación de la seleccionada directamente.
*acción:* En principio se marcará la opción 'Local x-Window Session'.

5. Configuración de x-Window:

Seleccionar el tipo de servidor x-Window deseado, estará determinado por el tipo de núcleo que se eligió inicialmente.

- Font server

Permite seleccionar un servidor de fuentes indicando su ruta y puerto de escucha.
*acción:* Se dejará la opción por defecto.

- Video Hardware

Indicar los drivers de tarjeta gráfica necesarios para el cliente.
*acción:* Se dejará la opción 'auto'.

- Video Modes

Selccionar aquellos modos de video que sean soportados por el cliente y el monitor del que disponga.
*acción:* Se dejará el modo 800x600.

- Colour Depth

Profundidad de color en bits. Indicar la correspondiente al hardware que se posea, cuanto mayor sea más recursos gráficos necesitará.
*acción:* Se dejará seleccionada la opción por defecto (24).

- Monitor Frequency

Frecuencia del monitor del cliente. La base de datos de monitores provee de frecuencias preconfiguradas.
Options
No acceleration: Desactivar la aceleración por hardware
No hardware cursor:
Don't zap:
Disable screensaver: Desactivar el salvapantallas

Extra options: Permite pasar parámetros opcionales a las X
*acción:* Se dejarán las opciones por defecto.

6. Configuración de XDM:

- Connection Method

Método de conexión que se usará para contactar con el servidor XDMCP. Hay tres posibles: Directo, Indirecto y Broadcast
Indicar situación del servidor y el puerto en aquellos casos que sea necesario.
Existe la posibilidad de preguntar estos valores una vez iniciado el cliente.
*acción:* Se seleccionará la opción 'Broadcast'.

7. Configuración de opciones generales:

Seleccionar la configuración deseada

- Enable debug

Activar debug
*acción:* No se seleccionará.

- Wait for key press before starting x windows

Esperar a que se presione una tecla antes de iniciar Xwindows
*acción:* No se seleccionará

- Wait for key press before connecting to login server

Esperar a que se presione una tecla antes de conectar al servidor de login.
*acción:* No se seleccionará.

- Wait for key press on error connecting to login server

Esperar a que se presione una tecla cuando suceda un error al conectar al servidor de login
*acción:* No se seleccionará.

- Enable login shell (no authentication)

Activar login por shell sin autentificación.
*acción:* No se seleccionará.

- Enable login shell (authenticated)

Activar login por shell con autentificación.
*acción:* No se seleccionará.

- Enable telnet server

Activar Servidor telnet.
*acción:* No se seleccionará.

- Set root password

Indicar el password de root.
*acción:* Se indicará un password genérico.

- Enable tiny web server

Activar miniservidor web.
*acción:* No se seleccionará

- Boot mensages

Indicar la cantidad de mensajes de boot deseada.
*acción:* Se dejará la opción por defecto.

- Remote Configurations

Permite activar la configuración remota del cliente. Se ha de indicar la ip del servidor servidor y el directorio en el que se encuentran los archivos.
*acción:* No se seleccionará.

## Referencias


- *ltsp* http://www.ltsp.org [1]
- *netstation* http://netstation.sourceforge.net [2]
- *pxes* http://pxes.sourceforge.net/ [3]
- *soporte hardware para pxes* http://pxes.sourceforge.net/readme.html#compliant%20hardware [4]
- *informe retrosystems* http://www.retrosystems.com/new_page_8.htm [5]
- *rom-o-matic* *http://www.rom-o-matic.net/5.2.5* [6]