El software libre en educación y sus aportes a la educación y formación constructiva en valores

Zulma Cataldi y Fernando J. Lage
Facultad de Ingeniería Universidad de Buenos Aires.
Facultad Regional Buenos Aires Universidad Tecnológica Nacional
Ciudad de Buenos Aires. Argentina
liema@fi.uba.ar; flage@fi.uba.ar

RESUMEN

A través del análisis de las características y ventajas del software libre y de código abierto se busca que los docentes puedan identificar sus necesidades educativas a fin encontrar productos alternativos aptos para sus programas educativos.
Si bien no quedan resueltas todas las necesidades con un solo paquete se puede recurrir a diferentes programas atendiendo a cada una de ellas; siempre a un costo menor y con una mayor vida útil de los recursos de hardware disponibles; entre otras posibilidades. La elección se debe acompañar de un cambio centrado en la libertad para investigar, crear, modificar y aprender basada en la colaboración.

Palabras clave:
Software libre, código abierto en educación

1. INTRODUCCIÓN

En América Latina se observa un incremento en la utilización de GNU[1]/Linux y según la compañía de estudios en tecnología IDC[2], se espera que la venta de paquetes de software GNU/Linux para computadoras de escritorio y servidores a nivel mundial siga en aumento debido a las empresas que adoptan software libre (SL) para administrar sus negocios debido a razones como: reducción de costos, estabilidad, flexibilidad, estándares abiertos, calidad y seguridad, independencia de plataforma, escalabilidad, mayor rendimiento, multiplataforma, independencia del proveedor, personalización de los sistemas, entre otras.
Esta tendencia se fundamenta en motivos:

- *Económicos*: El objetivo es reducir los costos de las licencias de ambos tipos de aplicaciones, por ejemplo que el sistema operativo sobre el cual corre la aplicación específica sea SL.
- *Legales*: Según informes de la ONG[3] Software Legal, los índices de piratería en Argentina alcanzan el 65 por ciento, por lo que se deberá prever la forma de regularizar esta situación por lo que una alternativa a considerar es el SL.
- *Estratégicos*: El SL se basa en la utilización de estándares abiertos para su desarrollo, esto garantiza dos condiciones: primero la interoperatibilidad entre sus aplicaciones y segundo y más importante la independencia en la elección de la aplicación.
- *Morales*: Copiar software es un delito aunque el objetivo sea para ayudar a alguien.

[1] GNU es un acrónimo recursivo para *"GNU No es Unix"*
[2] IDC sitio web www.idc.com
[3] ONG Organización No Gubernamental

2. ALGO DE HISTORIA DEL SL

Durante los años sesenta el software se consideraba un servicio, no un producto en sí mismo, y se lo concebía como un complemento necesario para poder utilizar las computadoras. En esa época era usual que los usuarios y los programadores compartieran libremente sus aplicaciones. Pero, hacia fines de los años setenta las compañías comenzaron a aplicar algunas restricciones a los usuarios debido a la implementación de las licencias, es decir, a través de contratos o acuerdos de uso entre el dueño del *copyright* y el usuario de la aplicación, en el cual se establecían los límites de uso, modificación y distribución. En 1985, Richard Stallman [1] creó la *Fundación del Software Libre*[4] (FSF, según sus siglas en inglés) y acuñó el concepto de *copyleft* como posible traducción: *"izquierdos de autor"* y dando una definición para el SL si garantiza las cuatro libertades básicas de la Tabla 1:

Tabla 1: Libertades básicas.

libertad 0:	La libertad de usar el programa, con cualquier propósito
libertad 1:	La libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades
libertad 2:	La libertad de distribuir copias, con lo que puedes ayudar a tu vecino
libertad 3:	La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie.

Cabe destacar que el cumplimiento de las libertades 1 y 3 implica el acceso al código de fuente del programa, lo que permite que cualquier programador con un editor de texto simple pueda entender su funcionamiento.
Considerando la ambigüedad del término *libre* (en inglés se considera *free* a todo aquello que es libre o gratis) es necesario aclarar que se refiere a "*libre*" a tener las libertades descriptas y ello no implica que sea gratuito necesariamente.
La contraposición a este modelo es el software privativo o propietario (SP), es decir, todo programa que prive a los usuarios de las libertades para usarlo, modificarlo y distribuirlo o que requiera que se solicite autorización para ello. Este término también es aplicable al software cuyo costo es tan elevado que no puede ser afrontado por un particular.
Hacia 1986 se puso en marcha el proyecto GNU, se buscaba concebir un sistema operativo versión libre como alternativa al Unix existente, que estuvo casi listo en 1989, y solo le faltaba el núcleo o kernel. Recién en 1991 se completó el sistema operativo GNU/Linux cuando Linus Torval liberó el *kernel*[5] Linux bajo la licencia GPL[6] y lo adoptó el proyecto GNU. Las siglas (GPL, GNU, etc.) vistas representan las distintas licencias que existen sobre el software libre. En la Tabla 2 se resumen las más populares en la actualidad.

Durante 1988 algunos miembros de la comunidad del *SL* comenzaron a utilizar la denominación *Código Abierto* (OSS, por su sigla en inglés) con el objeto de evitar la ambigüedad de la palabra libre (*free*) en inglés. Con el tiempo nació un nuevo movimiento que si bien es compatible con el SL, sus intereses finales difieren. El software de código abierto

[4] http://www.fsf.org/ *Definición del software libre*, disponible en: http://www.fsf.org/licensing/essays/free-sw.html, consultado el 10/11/07.
[5] es la parte fundamental de un sistema operativo.
[6] GPL es General Public License.

(OSS) es software cuyo código fuente está disponible públicamente, aunque los términos de licenciamiento específicos varían. Las licencias deben cumplir diez condiciones para ser consideradas licencias de OSS [2].

1. Libre redistribución: el software debe poder ser regalado o vendido libremente.
2. Código fuente: el código fuente debe estar incluido u obtenerse libremente.
3. Trabajos derivados: la redistribución de modificaciones debe estar permitida.
4. Integridad del código fuente del autor: las licencias pueden requerir que las modificaciones sean redistribuidas sólo como parches.
5. Sin discriminación de personas o grupos: nadie puede quedar afuera.
6. Sin discriminación de áreas de iniciativa: los usuarios comerciales no pueden ser excluidos.
7. Distribución de la licencia: deben aplicarse los mismos derechos a todo el que reciba el programa
8. La licencia no debe ser específica de un producto: el programa no puede licenciarse sólo como parte de una distribución mayor.
9. La licencia no debe restringir otro software: la licencia no puede obligar a que otro software que sea distribuido con el software abierto deba también ser de código abierto.
10. La licencia debe ser tecnológicamente neutral: no debe requerirse la aceptación de la licencia por medio de un acceso por clic del *mouse* o de otra forma específica del medio de soporte del software.

A partir de aquí surgen las dos vías conceptuales del SL. Por un lado, se encuentra la visión de la *FSF* en la cual el objetivo final está relacionado con el carácter moral y éste prima sobre la excelencia técnica del software. Y por otro, se sitúa la *OSI* (Open Source Initiative) donde compartir el código fuente es un medio para la búsqueda de la excelencia técnica. A pesar de las diferencias filosóficas de ambos movimientos pocas veces suele haber interferencia en el desarrollo y en la colaboración entre ambos proyectos. Sólo a los fines prácticos se llamará a ambas iniciativas con la denominación de SL con el objeto de facilitar la lectura.
Debido a que el SL permite el libre uso, modificación y redistribución, es recomendable su aplicación en los países del tercer mundo donde el costo de las licencias del software privativo es a veces inaccesible. También, brinda la posibilidad de modificarlo localmente y permite que sea posible su traducción a idiomas que no son necesariamente rentables comercialmente.

3. CARACTERÍSTICAS DEL SL PARA EDUCACIÓN

Las características del SL lo hacen propicio para ámbitos formativos, ya que los usuarios tienen la libertad de usarlo, ejecutarlo, copiarlo, distribuirlo ó mejorarlo sin necesidad solicitar permiso a nadie ya que goza de las libertades de la Tabla 1. Estas condiciones lo tornan ventajoso respecto del software propietario ya que en éste no hay autonomía para usarlo, copiarlo y redistribuirlo, el usuario no tiene el código fuente y las modificaciones que desee hacer deben ser solicitadas al propietario del mismo).
“La primera (razón para usarlo en las escuelas) es que el software libre supone un ahorro de costos para las escuelas. El software libre le da a las escuelas, igual que a cualquier otro usuario, la libertad de copiar y redistribuir el software, por lo que pueden hacer copias para todas las computadoras que tengan. En los países pobres esto puede ayudar a reducir la

brecha digital" [1]. El SL permite que los estudiantes puedan aprender cómo funciona y para aprender a escribir buen software, los estudiantes necesitan escribir y leer mucho código, leer y comprender programas reales y el software privativo no les permite aprender en este sentido. Las escuelas deberían decirle a sus alumnos si llevan software a la escuela, deben compartirlo con los demás niños, por ello el software que instalen debería estar disponible para que los alumnos lo copien, se lo lleven y lo redistribuyan tanto como quieran [1].

El SL, como se señaló, es una cuestión de libertad no de costos, ya que el usuario tiene la libertad de distribuir copias modificadas o no, ya sean gratis o cobrando por su distribución, de este modo no hay exigencias de pagar o pedir permisos. Harari [3] señala que una entidad puede continuar con la actualización de su software propietario o considerar la posibilidad de migrar sus sistemas a SL, para lo cual deberá analizar ciertos aspectos tales como los que se señalan en la Tabla 3.
El SL se puede conseguir en forma gratuita o pagando por su distribución. En el mercado existen diferentes precios y las instituciones pueden elegir la opción que más convenga. Se pueden realizar todas las copias que se requieran del mismo en diferentes computadoras, sin necesidad de pagar por ellas. No se necesita realizar actualizaciones y siempre se encontrará soporte técnico. El SL permitiría a las instituciones reutilizar hardware obsoleto, ya que existen más posibilidades para hacerlo.

El desafío que enfrentan las organizaciones al evaluar si un producto *Open Source* particular es factible o no, difiere significativamente del producto comercial análogo. El desafío de la organización en la elección de un paquete comercial, se centra en identificar cuál es el vendedor que le ofrece el producto más completo: es decir, software, soporte, entrenamiento, etc. Los usuarios de *Open Source*, a su vez deben localizar y evaluar los componentes individuales del producto para armar un paquete completo que cubra las necesidades de la organización

Tabla 2. Distintas licencias de software libre

Tipo de Licencia	*Característica*	*Principales representantes*
Licencia de software libre sin protección heredada	Se pueden crear una o varias obras derivadas de la original sin necesidad de respetar la licencia original, es decir, a partir de un producto con licencia libre se puede generar uno con licencia no libre.	Academic Free License Apache Software License BSD License, MIT License. NCSA Open Source License; W3C Software Notice and License.
Licencia de software libre con protección heredada	Todas las obras derivadas de la original tendrán algunas de las características de la licencia original, aunque no todas.	Common Public License GNU General Public License GNU Lesser General Public License Mozilla Public License
Licencias semilibres	Las aplicaciones serán siempre para obra final sin fines de lucro.	Para desarrollos de software individuales y por lo tanto se deben analizar las aplicaciones por separado.
Licencias semilibres Antagónicas	Existen restricciones a la cantidad de usuarios o licencias: por ejemplo para n usuarios con licencias libres y para n+1 aplican licencias no libres.	Para desarrollos de software individuales, se deben analizar las aplicaciones por separado. Un ejemplo de esto podría ser el MS SQL para un solo usuario que no posee cargo asociado, mientras que para más si.
Licencias no libres	Existen restricciones a su uso, copia, redistribución, etc.	Son la mayoría de las aplicaciones de software a medida que se desarrollan y también la de los "*enlatados*" más utilizados.

.

Tabla 3: Aspectos a considerar para la migración [3].

Aspectos	*Al migrar al software libre*	*Actualizando el software propietario*
Tecnológicos	Se dispone del código fuente, se puede adaptar y modificar de acuerdo a las necesidades propias y aporta crecimiento tecnológico para la entidad.	No se dispone del código fuente y las modificaciones hay que requerirlas al creador el software. El crecimiento tecnológico es de las la empresa creadora del software.
Formativos	La formación de los usuarios en el uso de este tipo de software es una inversión, los conocimientos que se adquieran servirán para el futuro. Los programas *no* cambian por cuestiones comerciales.	La formación de los usuarios en este tipo de software representan un gasto, los conocimientos adquiridos pueden no servir en el futuro. Los cambios, a veces superfluos, se realizan por cuestiones comerciales.
De actualización	La actualización no depende de intereses comerciales, sino de la necesidad de la entidad. Siempre habrá soporte técnico para las mismas.	Generalmente pasado cierto tiempo hay que realizar actualizaciones por más que no las necesite, dado a veces se deja de dar soporte técnico al software obsoleto.
De hardware	Al no estar obligado a hacer actualizaciones del software no está obligado a hacer actualizaciones del hardware. El software libre permite posibilidades de volver a aprovechar el hardware obsoleto.	Las actualizaciones de software normalmente implican actualizaciones de hardware, ya que requieren hardware más potente.
De estándares	El software libre se basa en estándares, eso le da la posibilidad al usuario de seleccionar un espectro de aplicaciones amplio, actualizando el software propietario.	El software propietario no se basa, por lo general, en estándares, por lo tanto, el usuario se encontrará que es un cliente cautivo. El usuario deberá moverse dentro de las limitaciones del software propietario.
De seguridad	Al tener acceso al código fuente el usuario puede saber lo que hace el programa y si es seguro.	Con el software propietario no se puede saber si es seguro o no
De compatibilidad	Al almacenar, siempre, los datos en formato estándar, por más que las versiones evolucionen se puede acceder a la información..	La información almacenada en formato propietario, en general no es compatible con versiones mas antiguas.
De costos	Se puede instalar el software, en todas las máquinas que quiera, ya que se tiene la libertad de realizar y redistribuir todas las copias que desee. El software puede no tener costo, o se puede pagar por una distribución. Se puede elegir el precio que más conviene.	Se deberá pagar la licencia de cada máquina dónde se quiera instalar el programa propietario y en general los costos son elevados.
De caducidad	Se puede utilizar la versión del software que quiera todo el tiempo que lo desee ya que se podrá contratar a cualquiera para que le de soporte. Las licencias que no tienen tiempo límite, generalmente, no son válidas para las versiones posteriores.	Pagar por una licencia, no implica que sea para siempre ni que con ella se pueda acceder a las diferentes versiones del software. Algunas licencias tienen un tiempo límite y vencido ese lapso, se deberá volver a pagar por las ellas.

4. SOFTWARE BÁSICO DISPONIBLE: LIBRE O DE CÓDIGO ABIERTO

Ahora, se verán algunas de las variantes que se pueden utilizar para armar una configuración de programas utilizando SL, pensado en una PC con recursos para un estudiante e inclusive un investigador.

Tabla 4: Diferentes software disponibles

Tipo de programa	*Objetivo*	*Software disponible en el mercado*	
Sistema Operativo	Es un conjunto de programas destinados a permitir la comunicación del usuario y la computadora y gestionar sus recursos. Comienza a trabajar cuando se enciende la computadora y gestiona el hardware de la máquina desde los niveles más básicos.	Free-BSD NetBSD Open-BSD FreeDOS Darwin, Linux	Minix OpenSolaris PcBSD Plan9 Reactos
Herramientas de Ofimáticas	No existe una norma estricta sobre los programas a incluir en una suite de ofimática, pero la mayoría incluyen un procesador de textos y una hoja de cálculo. La suite puede contener un programa de presentaciones, un sistema gestor de base de datos y herramientas menores de gráficos y comunicaciones. También pueden contener: un programa de organización (agenda), un navegador web y un cliente de correo electrónico.	Open Office Gnome Office StarOffice. Papyrus OFFICE	
Herramientas de manejo de imágenes	Permite la edición digital de fotografías y dibujos por medio de la computadora, pudiendo realizar correcciones, trabajo en capas, etc. para mejorar y resaltar todo tipo de imágenes.	GIMP	
Navegadores de Internet	Es un programa o grupo de programas que se utilizan para acceder a contenido HTML disponible en Internet. Puede estar complementado por otras aplicaciones, como gestores de correo electrónico, agendas, etc.	Firefox, Mozilla Mozilla Thunderbird, un cliente de correo electrónico Mozilla Sunbird, un programa de calendario	
Otras aplicaciones	Existen programas específicos para otras necesidades		

Para ello, en la Tabla 4 se resumen las principales aplicaciones que una computadora actual debería poseer para brindar a sus usuarios las funcionalidades básicas y extendidas para realizar sus tareas:

4.1. Open Office

Una de las aplicaciones de ofimática de *OSS* es *Open Office*[7] que es un conjunto de herramientas con una interface de usuario muy similar a la que brinda Microsoft Office e incluye varias aplicaciones:
– un procesador de texto: OpenOffice Writer
– un editor de presentaciones: OpenOffice Impress
– un editor de hojas de cálculo: OpenOffice Calc
– un graficador: OpenOffice Graph
– Para las aplicaciones de Dibujo: OpenOffice Draw

OpenOffice originalmente estaba basado en la suite *StarOffice*, desarrollada por StarDivision y adquirida por Sun Microsystems en agosto de 1999. El código fuente de la suite fue liberado en julio de 2000, con la intención de hacer frente al dominio en el mercado de Microsoft Office dando así, una alternativa abierta, de bajo costo y alta calidad y con un código disponible con licencia LGPL[8].

De manera paulatina, pero consistente, esta suite de escritorio fue ganando terreno en el mercado, ayudada por el hecho de que la mayoría de las distribuciones Linux del mercado para PC de escritorio, la traen instalada. En la Figura 1 se observa la pantalla del procesador de textos.

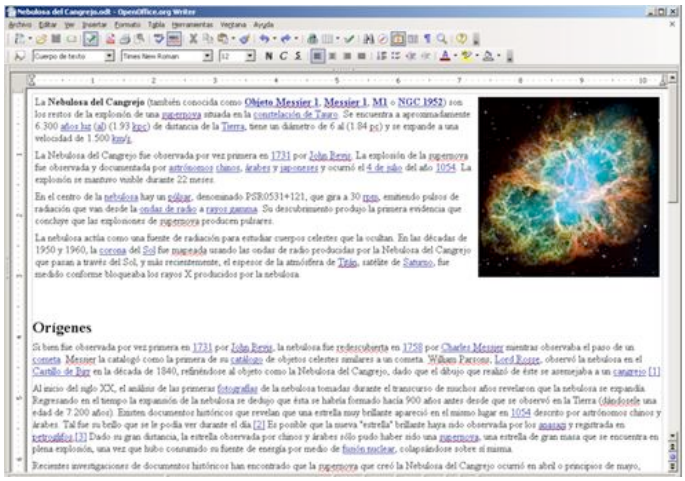

Figura 1: Pantalla del procesador de textos.

4.2. Software de manejo de imágenes

[7] http://www.openoffice.org
[8] LGPL es *Lesser General Public License* o *Licencia Pública General Menor.*

GIMP (GNU Image Manipulation Program) es un programa de manejo de imágenes del proyecto GNU que tiene licencia GNU (General Public License). GIMP sirve para procesar gráficos y fotografías digitales con usos típicos que incluyen la creación de gráficos y de logos, cambio de tamaño y recorte de fotografías, cambio de colores, la combinación de imágenes usando un paradigma de capas, la eliminación de elementos no deseados de las imágenes y la conversión entre distintos formatos de imágenes. También se puede utilizar GIMP para crear imágenes animadas sencillas y es la alternativa del SL al programa de retoque fotográfico Photoshop. La primera versión se desarrolló para sistemas Unix y fue pensada especialmente para GNU/Linux, sin embargo actualmente existen versiones totalmente funcionales para Windows y para Mac OS X.

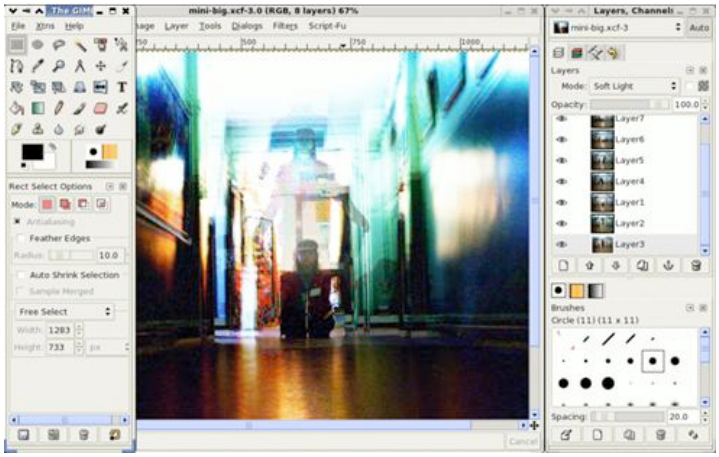

Figura 2: Pantalla de GIMP.

4.3. Acceso a Internet

Para el acceso a Internet el navegador web es *Mozilla* o *Firefox*[9], que fue desarrollado por la Corporación *Mozilla* y un gran número de voluntarios externos. *Firefox*, comenzó como un derivado del *Mozilla Application Suite* y terminó por reemplazarlo como el producto bandera del proyecto *Mozilla*. *Firefox* es un navegador web multiplataforma, que está disponible para Microsoft Windows, Mac OS X y GNU/Linux. Sin embargo el código ha sido portado a otros sistemas operativos como FreeBSD, OS/2, Solaris, SkyOS, BeOS y más recientemente a Windows.
El código fuente de Firefox está disponible bajo la triple licencia de Mozilla como un programa libre y de código abierto. En la Figura 3 se observa la página de Wikipedia, desde el navegador Mozilla Firefox.

4.4. Otras aplicaciones

En la actualidad existen aplicaciones libres de todo tipo que se utilizan para actividades variadas. A continuación se presentan algunas de ellas que pueden asistir a la educación:

– *Arduino*[10]: Es una plataforma *Open Source* de computación física basada en una simple placa de Entrada Salida (I/O: Input/Output) y un entorno que emplea el lenguaje de

[9] http://www.mozilla.org
[10] www.arduino.cc

programación Processing/Wiring. Es una herramienta orientada a enriquecer el uso tradicional de la tecnología y el proceso de comunicación que se puede utilizar para crear objetos interactivos autónomos, o para interactuar con otras aplicaciones de software como Flash.

Figura 3: Interface del navegador Mozilla Firefox.

- *Blender*[11]: Es un software de código fuente abierto para creación gráfica en 3D: diseño, animación, postproducción y creación interactiva. Actualmente es compatible con todas las versiones de Microsoft Windows, Linux, Solaris, FreeBSD, IRIX y MacOS X.
- *Criptored*[12]: En un software de criptografía de libre distribución publicado en el servidor de *CriptoRed* por diferentes autores en el sitio de la Universidad Politécnica de Madrid. El material docente y las aplicaciones para su desarrollo son de distribución libre.
- *EVE* (ellite veejay engine): Es un software de código abierto orientado a la mezcla de video en directo y programado con Pure Data en y para Linux donde toda su música tiene licencia Creative Commons
- *Jahshaka, Powering the New Hollywood:* Es un software de código abierto para postproducción audiovisual digital que permite la edición en tiempo real y la creación de efectos y es compatible con Linux, OsX, Iris, Windows, y Solaris.
- *Red Libre Red Visible*[13]: Este proyecto permite observar los flujos de información intercambiados mediante redes inalámbricas conectadas a Internet como medio para generar una comunidad para el acceso libre y gratuito a la red.

5. VENTAJAS DEL SL EN EDUCACIÓN

Las ventajas de índole práctica del SL derivan de su modo de producción que son las redes distribuidas entre pares que colaboran por diversos motivos a través de la transparentación del proceso de producción. Por otra parte, la propia filosofía seguida para su construcción es una manera efectiva de conseguir fiabilidad en el software publicando el código para que lo revisen otros programadores. Como consecuencia, el SL promueve la cooperación entre las personas donde el software privativo la convierte en un delito, siendo ésta un valor fundamental de la sociedad al que la escuela debe prestar especial atención.

[11] www.blender.org
[12] www.criptored.upm.es
[13] www.lalalab.org

Si bien algunos adolescentes no sienten curiosidad por saber cómo están hechos los programas de computadora, hay valores generales que persigue la educación que están en conflicto con el mensaje que transmite el software privativo. Las escuelas deben enseñar hechos, conceptos, principios y procedimientos, pero también valores y la misión de la escuela es enseñar a las personas a ser buenos ciudadanos, a cooperar con los demás, a ser solidarios siendo esta la base de la sociedad. Cooperar significa entre otras cosas, compartir software, poder hacer copias a todos los compañeros de clase, llevarse a casa el software que se usa en la escuela, pero esto, con el software privativo representa un delito [4].

Enseñar a los estudiantes a usar SL y a participar en la comunidad de usuarios y desarrolladores de software libre es una lección de ciudadanía llevada a la práctica que enseña a los estudiantes que el ideal es el modelo de servicio público y la solidaridad, no el modelo del beneficio a cualquier precio de las multinacionales. Todos los niveles pueden y deben usar software libre [1].
Belin y Heinz [5] enumeran lo que se enseña con el SL:

- *"Que no todo está hecho.*
- *Que aún hay retos y que las cosas siempre se pueden mejorar.*
- *A adoptar una postura constructiva.*
- *A cooperar con la comunidad local e internacional, sin distinción de edades, razas, nivel social, títulos, etc.*
- *(…) A propagar el conocimiento de forma libre (…)*
- *A trabajar en equipo.*
- *La libertad de investigar, crear, modificar y aprender".*

La filosofía del SL se resume en el mapa conceptual de la Figura 4 recreado de Mérou [6]. El autor buscó poner la menor cantidad de conceptos en el mapa para una interpretación más sencilla y poder hacerse una idea general. Por eso, ha incluido la cuarta libertad (poder redistribuir las modificaciones) dentro de la tercera. Los conceptos están en los recuadros, las relaciones sobre las flechas que los unen y los ejemplos o las notas sobre los conceptos están debajo de los cuadros [6].
Amatriain [7] resume la coincidencia entre los valores del SL y la educación: *"los valores que una institución educativa tendría que promover están muy relacionados con aquellos que promueve el software libre: libertad de pensamiento, expresión, igualdad de oportunidades, esfuerzo y beneficio colectivo en lugar del beneficio individual, etc. De hecho, la libertad puede que sea el valor más importante relacionado con' la educación: la educación sin libertad se convierte en adoctrinamiento".*

El SL, por su flexibilidad, facilitaría la formación basada en competencias genéricas, transferibles a otras situaciones y entornos, y el desarrollo de la capacidad seguir aprendiendo por su cuenta a lo largo de toda la vida de los estudiantes. Pero, se trata más de una cuestión de enfoque didáctico que de la naturaleza del software, ya que se puede formar de la misma forma al estilo: *"Qué tecla hay que apretar"* con SL. Los fines que persiguen las empresas no son los mismos que los de las escuelas, por lo que la alfabetización tecnológica va más allá de saber manejar una *suite* ofimática [4].

El SL provee un entorno donde los estudiante pueden: a) trabajar activamente para resolver problemas reales en entornos colaborativos, desafiantes y motivadores, b) los problemas serán reales y lo suficientemente complejos para requerir un amplio rango de habilidades, herramientas y aptitudes y c) necesitarán aprender como trabajar en dichos problemas [8]. "*Pasar de software propietario a software libre es mucho más que cambiar una plataforma informática. Si se es coherente con la filosofía de software libre, los principios de cooperación e investigación, deben estar presentes*[14]*"*.

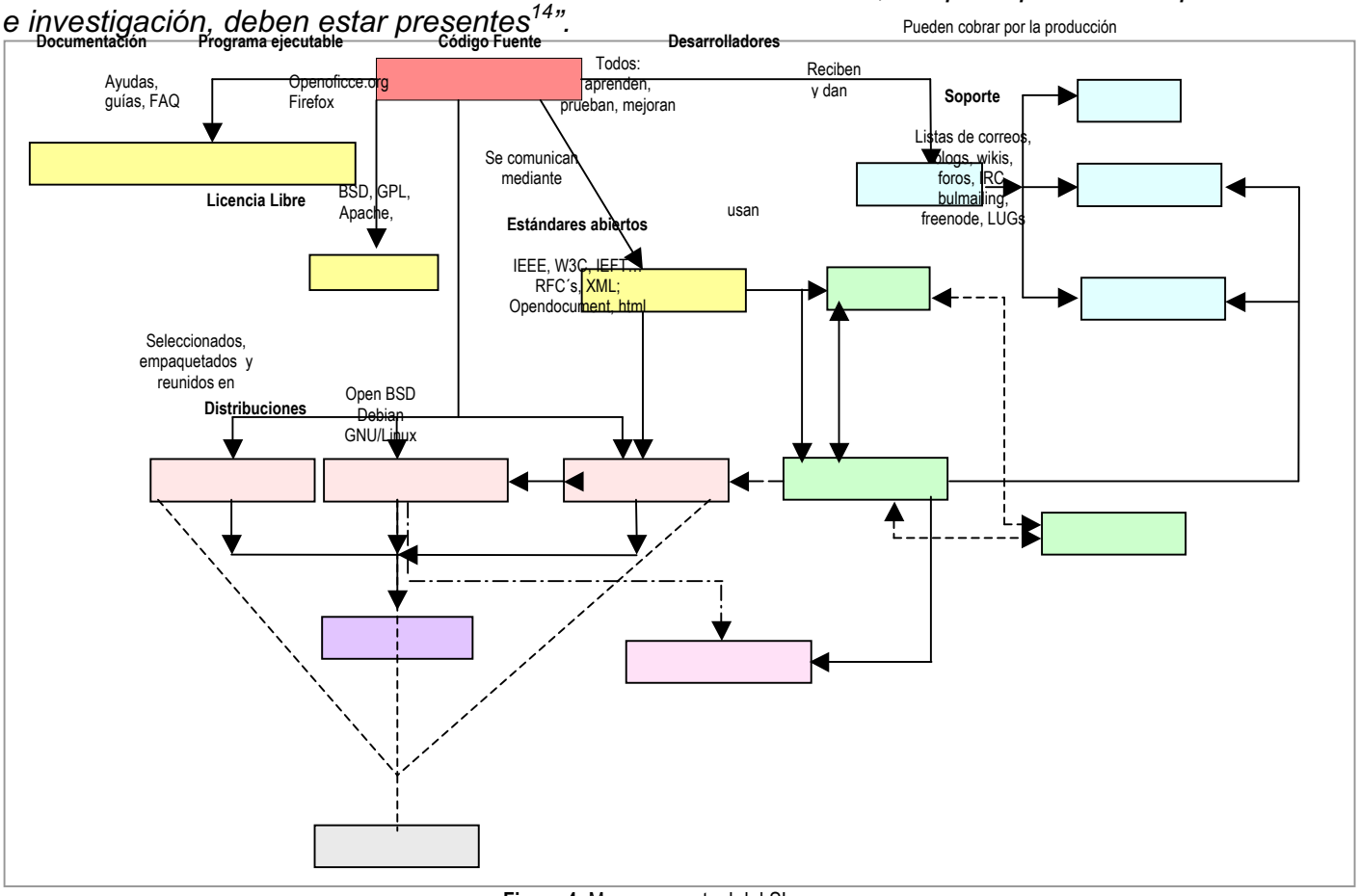

Figura 4: Mapa conceptual del SL.

Atwell [9] ha señalado que el SL en la educación se relaciona con la innovación educativa, por varias razones: a) en los proyectos de SL el costo inicial es bajo, b) suelen ser personales o de un pequeño grupo de entusiastas, c) se puede "construir" sobre el trabajo de otros proyectos y explorar sus aplicaciones educativas (por ejemplo integrando herramientas que originalmente no fueron diseñadas con propósito educativo, como *blogs* y *wikis*).

Se puede mencionar el caso de Moodle, la plataforma de enseñanza basada en presupuestos socio-constructivistas del aprendizaje que ha superado en funcionalidades a sus alternativas privativas y que se ha vuelto muy popular. Fue iniciado por Dougiamas [10], estaba descontento con el diseño y funcionamiento del software privativo equivalente de su

[14] Es un modelo apto para la enseñanza de informática. a)Se enseñan conceptos fundamentales que sirven como base para utilizar herramientas informáticas, b) la enseñanza depende de los fundamentos y no tanto de las herramientas, c) Se enseña de manera tal que lo visual sirva para acelerar los conceptos analíticos adquiridos, d) el software libre es accesible a todos. No dependemos de una empresa en particular, e) se basa en plantear soluciones a nuevos desafíos, f) se fomenta un modelo colaborativo , g) el modelo de desarrollo se basa en compartir el código fuente de los programas, h) se aplica el método científico a la informática

Universidad, por lo que "construyó" una plataforma para sus clases que hoy día se convirtió en una comunidad Moodle formada por desarrolladores y usuarios: Hoy días suman millones de usuarios: estudiantes y profesores que utilizan Moodle en sus clases presenciales, semi-presenciales o a distancia [4].

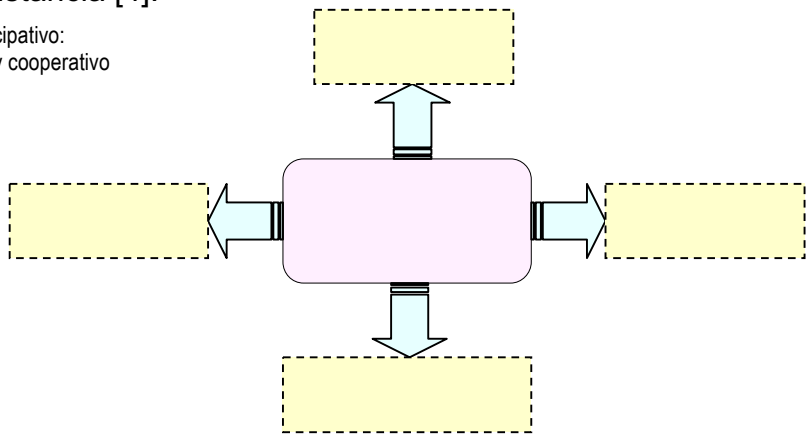

Figura 5: Ventajas del SL en educación.

Pero, el fuerte de Moodle son los cursos que contienen actividades y recursos disponibles (foros, glosarios, wikis, tareas, cuestionarios, encuestas, bases de datos, etc...) que se pueden adaptar y su potencia está en la combinación de la actividades en secuencias lo que permite guiar a los participantes a través de caminos de aprendizaje. Pero, no es el software en si mismo sino el uso y la pedagogía detrás del mismo que guía las actividades.

6. CONCLUSIONES

Existe una gran diversidad de aplicaciones de SL y abierto y el paquete de programas se debe armar de acuerdo a las necesidades de cada proyecto educativo. El SL se puede conseguir en forma gratuita o pagando por su distribución. En el mercado existen diferentes precios y se puede elegir la opción más conveniente y se pueden hacer las copias que se requieran del mismo en diferentes computadoras, sin necesidad de pagar por ellas.

En general *Open Office* resuelve el problema de las herramientas ofimáticas con una interface de usuario muy similar a la que brinda Microsoft Office e incluye aplicaciones similares. Las necesidades de acceso a Internet deben cubrirse con Mozilla Firefox y para otras que surjan deberán buscarse específicamente.

Las condiciones del SL lo tornan ventajoso respecto del software propietario ya que en éste no hay autonomía para usarlo, copiarlo y redistribuirlo porque el usuario no tiene el código fuente y las modificaciones que requiera hacer deben ser solicitadas al propietario del mismo.
El SL ofrecer grandes ventajas a la comunidad de la educación de la informática: a) desarrollar (o fomentar la existencia del desarrollo) software que puede se usado y mejorado por una comunidad internacional, b) provee un laboratorio de tamaño mundial, gente y otorga la experiencia en la colaboración y el desarrollo en grandes productos de software, c) permite obtener evaluación sobre el software desarrollado y d) amplía la metodología por la cual se aprende, se aplica y se enseña la informática [8].

"Así como para un científico es beneficioso asegurar que cualquiera pueda entender, modificar y aplicar sus ideas. Para un informático (como científico) el Software Libre le asegura que cualquiera tendrá acceso a su código" [8].

7. AGRADECIMIENTOS

Este artículo es parte del PID: *Modelado del tutor basado en redes neuronales para un sistema tutor inteligente.* SeCyT 2007-2008. Universidad Tecnológica Nacional. Facultad Regional Buenos Aires. EZINBA639. Acreditado en el Programa Incentivos Código 25/C099. Convenio UTN-FRBA y FI-UBA.

8. REFERENCIAS

[1]. Stallman, R. (2004) *Porqué las escuelas deberían usar software libre?* http://www.gnu.org/philosophy/schools.es.html traducción del 23 julio de Miguel Abad Pérez
[2]. Open Source initiative (2005), *Definición del software de código abierto*, disponible en: http://www.opensource.org/docs/definition.php, consultado el 27 de Mayo de 2005, 20:12.
[3]. Harari, I. (2006) *Software Libre en las Escuelas*. Facultad de Informática: UNLP.
[4]. Adell Segura, J. y Bernabé-Muñoz, Y. (2007) *Software libre en educación* (Cap. 11) en Cabero (2007) *Tecnología Educativa.* Mc. Graw Hill.
[5]. Belin, S. y Heinz, F. (2006) *Software Libre y Software Privativo: Dos modelos de enseñanza.*
[6]. Mérou, R. (2005) *Mapa conceptual del software Libre.* http://es.gnu.es/~remene/map/map-es.png. Consultado 25-10-07.
[7]. Amatriain, X. (2004) *Free software in education. A guide for its justification and implementation.* Disponible en http://www.iua.upf.edu/~xamat/FreeSoftware/ProgramariLliureEducacio/ProgramariLliureEducacio.html consultado el 29-011-07
[8]. de Castro, M. V. (2003) *Software libre en Educación.* Doctorado en Informática y Modelización Matemática. Universidad del Rey Juan Carlos.
[9]. Atwell (2005) *What is the significance of Open Source for Education and training Community?* Proceedings of The First International Conference on Open Source Systems. Genova 11-15 julio.
[10]. Dougiamas, M. (2004) Videoconferencia: *Moodle in the future*. MoodleMot Spain 2004. 13 de setiembre. Centre d'Educació i Noves Tecnologies de la Universitat Jaume I de Castelló. Disponible en http://cent.uji.es/pub/node/245