¿QUÉ SISTEMAS DE AUTOMATIZACIÓN CONSIDERA ADECUADOS PARA ORGANIZAR UNA BIBLIOTECA JURÍDICA?

Óscar ARRIOLA NAVARRETE*

SUMARIO: I. *Características*. II. *Software propietario*. III. *Definición*. IV. *Características*. V. *Ventajas y desventajas*. VI. *Reflexiones finales*. VII. *Bibliografía*.

Las tecnologías de información y comunicación (TIC) están inmersas en muchas actividades humanas. Su uso y aprovechamiento son cada vez más comunes. Se busca aprovechar los beneficios que brindan las TIC para mejorar el entorno en el que vivimos, ya que con su buen uso se mejoran muchos procesos y servicios.

En la actualidad, los desarrollos tecnológicos han sido explotados por instituciones, empresas, organizaciones, etcétera, con la finalidad de estar en continua actualización. Las bibliotecas también tienen que estar en constante evolución, aprovechando las TIC, que permiten estar a la vanguardia, y de esta forma brindar mejores servicios. Es por eso que el mercado de *software* para bibliotecas ha crecido en los últimos años de manera exponencial, y existen actualmente una gran diversidad tanto de sistemas integrales de automatización de bibliotecas (SIAB) como de sistemas de descubrimiento.

En la era digital, en que los servicios y datos se basan en la Web y en la computación en nube, la tecnología utilizada por las bibliotecas se ha mantenido en arquitecturas ya caducas, como la cliente-servidor. La infraestructura tecnológica que las bibliotecas implementarán en los próximos años puede afectar a su capacidad para gestionar las operaciones internas de manera eficiente y proporcionar servicios de calidad en persona y en línea. Los productos tecnológicos débiles u obsoletos comprometen el futuro.

* Profesor titular "C" de tiempo completo en la Escuela Nacional de Biblioteconomía y Archivonomía.

De este modo, las bibliotecas tendrán que seleccionar aquellos productos que mejor se alinean con sus estrategias, con el servicio que proporcionan a sus usuarios, y aquellos que faciliten el trabajo de su personal.[1]

Al existir tal variedad de *software* especializado para bibliotecas, arroja la pregunta ¿cuál es el más adecuado para mi biblioteca? Los bibliotecarios tienen que tomar en cuenta las necesidades y objetivos de la biblioteca, así como también el presupuesto para determinar cuál es el sistema más adecuado. No siempre el *software* más avanzado y caro es el mejor; lo más importante es que cubra la mayor parte de las necesidades de la biblioteca, y, por consiguiente, ayude a mejorar los procesos y servicios. Es importante también que cuente con la posibilidad de actualizarse cada vez que se requiera y que tenga un soporte técnico eficiente.

La demanda de información es cada vez mayor, pues los usuarios reclaman servicios de información ágiles y de calidad; para que las bibliotecas puedan brindar estos servicios necesitan explotar lo más posible los beneficios que brinda un SIAB o su sistema de descubrimiento.

I. Características

Independientemente de la opción que se utilice para automatizar una biblioteca, ya sea *software* propietario o *software* libre, éste debe tener ciertas características que garanticen que es confiable su uso. Algunas de las características que debe tener el *software* son[2] (Open Source Software. 2005):

a. *Fiabilidad*: se define como el tiempo que un sistema puede permanecer en operación sin intervención del usuario.
b. *Calidad:* comúnmente se define como el número de errores en un número fijo de líneas de código.
c. *Seguridad:* lo resistente que es el *software* para no autorizar acciones fuera de protocolo (por ejemplo, virus).
d. *Flexibilidad:* la facilidad con que el *software* puede ser personalizado para satisfacer las necesidades específicas, y que se pueden ejecutar en diferentes tipos de dispositivo.
e. *Gestión de proyectos*: la facilidad de organizar los proyectos en desarrollo.

[1] Breeding, Marshall, "Library Systems Report 2016: Power Plays" *American Libraries: the Magazine of the American Libraries Association,* vol. 47, núm. 5, mayo de 2016, pp. 30-43.

[2] "Open Source Software", *Postnote*, núm. 242, junio de 2005, disponible en: *http://www.parliament.uk/documents/post/postpn242.pdf*.

f. *Estándares abiertos*: los documentos creados con un tipo de *software* deben ser leídos y trabajados en cualquier *software*.
g. *Los costos de cambio*: el costo de pasar de un sistema a otro.
h. *Costo total de propiedad*: la totalidad de los gastos durante la vida útil del *software*.
i. *Facilidad de uso:* lo fácil y amigable que es usar el *software*.

II. Software propietario

Es de gran importancia conocer el *software* propietario, ya que es una opción disponible en el mercado de la industria de la información, el cual se conoce también como *software* comercial. A continuación, se brindarán definiciones y una descripción de las características generales que lo representan.

III. Definición

En la actualidad, la tecnología se ha encargado de transformar y mejorar las bibliotecas o unidades de información, desde la forma que manejan y organizan el conocimiento hasta los servicios que prestan, ya que ahora tienen mayor alcance gracias al aprovechamiento de Internet. Estos recursos tecnológicos disponibles deben ser aprovechados por las bibliotecas. Para esto es necesario que se cuente con un sistema integral de automatización que ayude a este objetivo. Dentro del mercado actual existe una opción para la gestión de la información; es el llamado *software* propietario. A continuación, se verán algunas definiciones, así como también las raíces del término, que aportarán elementos para tener un concepto más claro de lo que es el *software* propietario.

El *software* propietario se usa como sinónimo de *software* no libre, también es llamado *software* privativo, *software* privado o *software* con propietario:

> Se refiere a cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo (con o sin modificaciones), o que su código fuente no está disponible o el acceso a este se encuentra restringido. En el *software* no libre una persona física o jurídica (por nombrar algunos: compañía, corporación, fundación) posee los derechos de autor sobre un *software* negando o no otorgando, al mismo tiempo, los derechos de usar el programa con cualquier propósito; de estudiar cómo funciona el programa y adaptarlo a las propias necesidades (donde el acceso

> al código fuente es una condición previa); de distribuir copias; o de mejorar el programa y hacer públicas las mejoras (para esto el acceso al código fuente es un requisito previo). De esta manera, un *software* sigue siendo no libre aun si el código fuente es hecho público, cuando se mantiene la reserva de derechos sobre el uso, modificación o distribución (por ejemplo, el programa de licencias *shared source* de Microsoft). No existe consenso sobre el término a utilizar para referirse al opuesto del *software* libre.[3]

La Free Software Foundation define al *software* propietario como aquel *software* que no es libre. Su uso, redistribución o modificación están prohibidos, o requieren que se solicite autorización; están tan restringidos que no puede hacerlos libres de un modo efectivo. Es de dominio privado, porque una determinada persona tiene la titularidad de los derechos de autor y goza de un derecho exclusivo respecto de su utilización. Niega a otras personas el acceso al código fuente del *software* y derecho a copiar, modificar y estudiar el *software*.[4]

Al respecto, Stella Rodríguez menciona que el *software* propietario es aquel *software* que está siendo desarrollado por una entidad que tiene la intención de hacer dinero del uso del *software*. El software propietario es aquel *software* que es imposible de utilizar en otro *hardware* o terminal, modificar y transferir sin pagar derechos a su creador o desarrollador.[5]

Retomando lo dicho anteriormente, se puede concluir que el *software* propietario es aquel que es de uso restringido, sólo para aquellos que paguen una licencia, aunque ésta no les da derecho a modificarlo, estudiarlo o explotarlo económicamente por cualquier medio, ya que está protegido por *copyright*, llevando la titularidad del dueño o creador, ya sea una persona física o una organización o corporación.

Es importante mencionar que el sistema de *copyright* funciona mediante la concesión de privilegios, y por lo tanto de beneficios a los editores y a los autores de una obra, llámese libro, disco musical, *software*, etcétera. El derecho de autor y *copyright* constituyen dos conceptos sobre la propiedad literaria y artística. La protección del *copyright* se limita estrictamente a la obra, sin considerar atributos morales del autor en relación con su obra,

---

[3] Culebro Juárez, Montserrat, Software *libre vs.* Software *propietario: ventajas y desventajas*, 2006, p. 17, disponible en: *http://www.softwarelibre.cl/drupal//files/32693.pdf*.

[4] Carranza Torres, Martín, *Problemática jurídica del* software *libre,* Argentina, Lexis-Nexis, 2004, p. 103.

[5] Rodríguez, Gladys Stella, "El *software* libre y sus implicaciones jurídicas", *Revista de Derecho,* núm. 30, 2008.

excepto la invención; no lo considera como un autor propiamente, pero tiene derechos que determinan las modalidades de utilización de una obra, a diferencia del derecho de autor.[6]

Dentro del ámbito bibliotecario se puede decir que al adquirir un SIAB que pertenezca a la categoría de *software* propietario es una buena opción que posee muchas ventajas, como es el soporte técnico, aunque también representa un costo importante, que no todas las bibliotecas pueden cubrir.

IV. Características

Existen grandes diferencias entre el *software* libre y el *software* propietario. Este último tiene ciertas particularidades que lo representan y que deben tomarse en cuenta al pretender implementarlo. Algunas de sus características son:

a. *Limitación de uso*: en el cual se debe tener licencias *staff*, usuarios u OPAC; estas licencias tienen que ser concurrentes (liberadas), puesto que poseen una vigencia, y, por tanto, si no se pagan no hay actualizaciones.
b. *Modificación:* al no conocer y no poder cambiar el código fuente del sistema adquirido, no se puede realizar una parametrización de acuerdo con las necesidades de la unidad de información; es de resaltar que esto sólo se puede realizar hasta donde lo permita el *software*.
c. *Distribución:* por ser un *software* propietario perteneciente a una entidad, se deben adquirir licencias para su uso, lo cual significa que no puede ser distribuido en diferentes equipos si no va acompañado de su licencia correspondiente. Por ser propietario y encontrarse protegido por *copyright*, el *software* no viene acompañado del código fuente al momento de adquirirlo.
d. *Costos:* en el caso de los costos, se debe tomar en cuenta el pago por adquisición inicial del *software*, el mantenimiento anual, la parametrización (cada que se requiera alguna adecuación), la capacitación sobre el uso y manejo al personal de la unidad de información; habrá que contemplarlo en el presupuesto que se posee.
e. *Adjudicación directa:* la mejor forma de adquirir el *software* es por un proceso de licitación.

[6] Culebro Juárez…, *op. cit.*

f. *Espacio en su servidor*: verificar qué tipo de servidor se posee, cuál es el espacio que se tiene para instalarlo, si se cuenta con un servidor espejo, que es una forma de almacenar o respaldar la información.
g. *Revisar si el proveedor es el desarrollador*:
h. *Ver el tamaño de la colección*: para adquirir un *software* se necesita comprobar si es adecuado al tamaño de la colección que se tiene.
i. *Proceso-licitación*

Saber qué es lo que se quiere:

— Linux
— Unix
— Postgrest
— Apache

Qué módulos se necesitan y cuáles ofrece el *software*:

— Adquisición
— Circulación
— Publicaciones periódicas
— Catalogación
— Administración

V. Ventajas y desventajas

1. *Ventajas*

Como ya se ha dicho, existe una gran diversidad de SIAB en el mercado actual, así que al tomar la decisión de implementar un *software* propietario a pesar del costo que representa, también es importante destacar todas las ventajas que se obtendrán al elegir esta opción para automatizar la unidad de información.

Algunas de las ventajas del *software* propietario mencionadas por Montserrat Culebro son:

a. *Control de calidad.* Las compañías productoras de *software* propietario por lo general tienen departamentos de control de calidad que llevan a cabo muchas pruebas sobre el *software* que producen.

b. *Recursos a la investigación.* Se destina una parte importante de los recursos a la investigación sobre los usos del producto.
c. *Personal altamente capacitado.* Se tienen contratados algunos programadores muy capaces y con mucha experiencia.
d. *Uso común por los usuarios.* El *software* propietario de marca conocida ha sido usado por muchas personas, y es relativamente fácil encontrar a alguien que lo sepa utilizar.
e. *Software para aplicaciones muy específicas.* Existe *software* propietario diseñado para aplicaciones muy específicas que no existe en ningún otro lado más que con la compañía que lo produce.
f. *Amplio campo de expansión de uso en universidades.* Los planes de estudio de la mayoría de las universidades de México tienen tradicionalmente un marcado enfoque al uso de herramientas propietarias, y las compañías fabricantes ofrecen a las universidades planes educativos de descuento muy atractivos.
g. *Difusión de publicaciones acerca del uso y aplicación del* software. Existe una gran cantidad de publicaciones, ampliamente difundidas, que documentan y facilitan el uso de las tecnologías proveídas por compañías de *software* propietario, aunque el número de publicaciones orientadas al *software* libre va en aumento.

2. *Desventajas*

Es importante revisar también la contraparte de los beneficios, es decir, las desventajas que puede tener el *software* propietario. Algunas contras que tiene el *software* propietario son:

a. *Cursos de aprendizaje costosos*. Es difícil aprender a utilizar eficientemente el *software* propietario sin haber asistido a costosos cursos de capacitación.
b. *Secreto del código fuente*. El funcionamiento del *software* propietario es un secreto que guarda celosamente la compañía que lo produce. En muchos casos resulta riesgosa la utilización de un componente, que es como una caja negra, cuyo funcionamiento se desconoce, y cuyos resultados son impredecibles. En otros casos es imposible encontrar la causa de un resultado erróneo, producido por un componente cuyo funcionamiento se desconoce.
c. *Soporte técnico ineficiente*. En la mayoría de los casos el soporte técnico es insuficiente o tarda demasiado tiempo en ofrecer una respuesta

satisfactoria, y puede ser legal o costosa la adaptación de un módulo del *software* a necesidades particulares.

d. *Es ilegal extender una pieza de* software *propietario para adaptarla a las necesidades particulares de un problema específico*. En caso de que sea vitalmente necesaria tal modificación, es necesario pagar una elevada suma de dinero a la compañía desarrolladora, para que sea ésta la que lleve a cabo la modificación a su propio ritmo de trabajo y sujeto a su calendario de proyectos.
e. *Derecho exclusivo de innovación.* La innovación es derecho exclusivo de la compañía desarrolladora. Si alguien tiene una idea innovadora con respecto a una aplicación propietaria, tiene que elegir entre venderle la idea a la compañía dueña de la aplicación o escribir desde cero su propia versión de una aplicación equivalente, para una vez logrado esto poder aplicar su idea innovadora.
f. *Ilegalidad de copias sin licencia para el efecto.* Es ilegal hacer copias del *software* propietario sin antes haber contratado las licencias necesarias.
g. *Imposibilidad de compartir*. Si una dependencia de gobierno tiene funcionando exitosamente un sistema dependiente de tecnología propietaria, no lo puede compartir con otras.
h. *Quedar sin soporte técnico*. Si la compañía desarrolladora del *software* propietario se va a bancarrota, tanto el soporte técnico como la posibilidad en un futuro de tener versiones mejoradas de dicho *software* y la posibilidad de corregir los errores de dicho *software* desaparecen. Los clientes que contrataron licencias para el uso de ese *software* quedan desprotegidos.
i. *Descontinuación de una línea de* software. Si una compañía desarrolladora de *software* es comprada por otra más poderosa, es probable que esa línea de *software* quede descontinuada y nunca más en la vida vuelva a tener una modificación.
j. *Dependencia a proveedores*. En la mayoría de los casos el gobierno se hace dependiente de un solo proveedor.
k. *Nulificación de desarrollo tecnológico de la industria nacional*. Nulidad de desarrollo tecnológico de la industria nacional respecto de la extranjera (las aplicaciones de consumo masivo se desarrollan en otros países).

Después de un atisbo general a los sistemas de automatización de bibliotecas de tipo propietario o comercial más instalados en México, se encuentra una alternativa más que está rompiendo los paradigmas con respecto a

la adquisición de un *software* para las diferentes unidades de información, independientemente del uso y las necesidades que éstas requieran.

Este nuevo concepto tiene ya un tiempo considerable a nivel internacional, primordialmente en los países desarrollados o con una amplia capacidad de adquisición de nuevas tecnologías de información. Dentro de la forma contemporánea de gestión de bibliotecas o de cualquier organismo encargado de preservar y difundir información se encuentra esta nueva tendencia, a la cual se le conoce como "*software* libre" o de "*open source*", que se explicará a detalle en el siguiente apartado.

3. *Software libre (free software) o de código fuente abierto (Open Source)*

Un *software* libre es todo aquel programa informático en el cual los que lo adquieren tienen la posibilidad de modificarlo y mejorarlo de la manera que más convenga; es decir, una vez obtenido el programa, éste puede ser ejecutado, cambiado, copiado, mejorado, modificado, usado, estudiado y distribuido libremente.

El *software* gratuito (denominado usualmente *freeware*) incluye en algunas ocasiones el código fuente; sin embargo, este tipo de *software* no es libre en el mismo sentido que el *software* libre, al menos que se garanticen los derechos de modificación y redistribución de dichas versiones modificadas del programa.

Con el *software* libre no es necesario solicitar ninguna licencia y los derechos de explotación son para toda la humanidad, porque pertenece a todos por igual. Cualquiera puede hacer uso de él, siempre con fines legales y consignando su autoría original.

Los dos anteriores conceptos proporcionan un claro ejemplo de la complejidad del término utilizado en inglés para el "*free software*", ya que *free* suele traducirse como libre o gratuito, y es aquí donde se desprenden las dos corrientes antes mencionadas, entre el *software* libre y el *software* gratuito, quedando esto claro, y para sustentarlo en una sola línea de investigación, se considerará únicamente como término genérico e incluyente *software* libre para efectos de este trabajo.

4. Definición

El término "*software* libre" y "*open source*" comparten modelos de desarrollo similares; sus principales diferencias se encuentran en sus aspectos

filosóficos. El software libre se enfoca en las libertades filosóficas que les otorga a los usuarios, mientras que el *open source* se enfoca en las ventajas de su modelo de desarrollo.

En este trabajo los tomaremos como sinónimos a ambos, aunque más bien sean complementarios.

Una definición que mejor describe al *software libre* la aportan Da Rosa y Heinz: El *software* libre se define por su tipo de licenciamiento. Por lo que se puede entonces llamar "*software* licenciado bajo condiciones libres". Simplificando al máximo, se debe entender que *software* libre es un *software* o programa de computación cuya licencia permite ejercer una serie de libertades.[7]

Así pues, el *software* libre es una fuente abierta que permite trabajar bajo cuatro libertades esenciales, esto dicho y comprobado por Richard Stallman, programador estadounidense y fundador del movimiento del *software* libre, quien en 1985 acuñó el término, y lo describe bajo estas condiciones:

— *Libertad de ejecutar* el programa sea cual sea el propósito (libertad 0).
— *Libertad de estudiar* cómo funciona el programa para ajustarlo a tus necesidades (libertad 1). (Es indispensable tener acceso al código fuente).
— *Libertad de redistribuir* copias, colaborando con otras personas (libertad 2).
— *Libertad de modificar*, de tal forma que la comunidad pueda aprovechar las mejoras (libertad 3). (Es indispensable tener acceso al código fuente).[8]

Wayner afirma que Stallman numeró las libertades empezando por el cero, porque así era como lo hacían los informáticos. Alguien calculó que era más sencillo empezar a numerar las bases de datos con el cero porque no se tiene que restar 1 tan a menudo.[9]

La Open Source Initiative utiliza la definición de Open Source para determinar si una licencia de *software* de computadora puede o no conside-

[7] Da Rosa, Fernando y Heinz, Federico,. *Guía práctica sobre* software *libre su selección y aplicación local en América Latina y el Caribe*, Organización de las Naciones Unidas para la Educación la Ciencia y la Cultura, 2007, disponible en: *http://unesdoc.unesco.org/images/0015/001560/156096s.pdf.z dtn nh57ujm*.

[8] GNU Operating System: Free Software Foundation, 2009, disponible en: *http://www.gnu.org/philosophy/free-sw.html*.

[9] Wayner, Peter, *La ofensiva del* software *libre. Cómo Linux y el movimiento del* software *libre se impusieron a los titanes de la alta tecnología*, Barcelona, Garnica, 2001, p. 129.

rarse *software* abierto. La definición se basó en las Directrices de *software* libre de Debian, que fue escrita y adaptada primeramente por Bruce Perens. Es similar, pero no igual a la definición de licencia de *software* libre.

Las licencias *open source* deben cumplir diez premisas para ser consideradas como tales:

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 dejarse fuera.
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 algún 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 de ratón o de otra forma específica del medio de soporte del software.

Éstas son características ineludibles en el contenido de las propiedades que hacen del *software* libre y del *open source* un elemento de estudio para su incorporación en cualquier sector de la sociedad; gracias a la acción cooperativa que posibilita el manejo de estos tipos de *software*, es que ha sido posible generar productos finales de gran envergadura y excelentes cualidades técnicas.

Los movimientos de *software* libre y del *open source* son de los más exitosos a nivel mundial en los últimos veinticinco años, impulsados por una comunidad internacional de programadores, con ética dedicada a la causa de la libertad y la cooperación.

El *software* libre presenta la ventaja de la independencia frente a vicisitudes y arbitrariedades en cuanto a las estrategias comerciales y a la continui-

dad de diversas herramientas y formatos que se utilicen para el tratamiento de la información en soporte electrónico.

La utilización de un *software* impacta en tres aspectos:

a. En el acceso a los servicios que ofrece una biblioteca o unidad de información,
b. En los documentos disponibles en soporte electrónico, y
c. A los programas y aplicaciones usados por la unidad de información para sus fines y mejoramiento de sus servicios.

La iniciación del *software* libre surgió debido a la fusión de varios movimientos liberales en torno al ámbito de la informática, que si bien esto ya se venía desarrollando años atrás y no fue hasta el periodo de 1970 cuando logró su mayor desarrollo con la creación de sistemas de código abierto o también conocido como *software* libre, estos factores actuaron de manera decisiva para el perfeccionamiento de dicho sistema; entre ellos se encuentran los *hackers*, el Internet, la filosofía de uso, así como también el flujo masivo de la información entre computadoras.

El uso de Internet fue vital para la evolución de los sistemas de código abierto, ya que en cuanto uno de sus sistemas es publicado en la red, es posible que alguien inmediatamente lo vea, lo instale y lo mejore. Existen diversos factores por los que sucede lo anterior, pero el que destaca un poco más es el de los *hackers* (principales promotores y desarrolladores de este tipo de *software*). Los *hackers* son los que promueven y definen la libertad del acceso a la información; sin embargo, la sociedad los tacha duramente, pues se tiene una idea falsa de que la primordial actividad de ellos radica en ingresar a las bases de datos de cualquier parte del mundo alterando sus códigos de acceso, con el propósito de robar y falsificar la información que contienen.

Por otra parte, el documento de Martín Carranza llamado "Problemática jurídica del *software* libre", presenta una reseña positiva referente a este *software*, manifestando que el *software* libre es más antiguo que el propietario, ya que como se explica anteriormente, en los años sesenta y setenta no se contemplaba como un producto, sino un añadido que los vendedores de grandes computadoras auxiliaban a sus clientes para que éstos pudieran usarlos. En dicha cultura era habitual que los programadores y desarrolladores de *software* compartieran libremente sus programas unos con otros. Este procedimiento era exclusivamente tradicional en algunos de los mayores grupos de usuarios de la época, como DECUS (grupo de usuarios de computadoras DEC).

En un inicio, al *software* libre no lo llamaban como tal; simplemente lo conocían como *software*, y con el paso del tiempo y con los grandes avances tecnológicos, este contexto tuvo cambios drásticos en los cuales las computadoras se modernizaron y utilizaban sus propios sistemas operativos propietarios, por lo que decidieron crear un *software* libre.

El *software* libre ha tenido un avance fundamental, lo cual ha logrado tener un papel relevante en el crecimiento y extensión de la red; esto es debido a que la mayor parte de la infraestructura del Internet se basa en protocolos abiertos.

A finales de los setenta, las compañías iniciaron la moda de aplicar limitaciones a los usuarios, con el uso de convenios de licencia.

Se cree que con la libertad de expresión se crea el *software* libre y se relaciona con el proyecto conocido como el proyecto GNU,[10] que fue fundado por Free Software Foundation (FSF), y escribe el acta fundacional de la comunidad: el Manifiesto GNU. Dicho proyecto tenía el objetivo de hacer un sistema operativo de forma que nadie tuviera que costear por el *software* y organizar una comunidad a partir del *software*.

En el año de 1971, mientras estudiaba en el primer año de la carrera de física en la Universidad de Harvard, Richard Stallman ingresó al Laboratorio de Inteligencia Artificial del Massachussets Institute of Technology (MIT), lo cual provocó que se convirtiera en un *hacker* del mismo laboratorio; en este laboratorio enseñaba a compartir el código fuente del *software* que se manejaba.

Stallman, disconforme con la idea de que el *software* tuviera propietarios, en ese año decide retirarse de su trabajo en el MIT con la finalidad de erigir y ampliar un sistema operativo perfecto.

Con el paso del tiempo, Stallman decidió implantar un sistema simultáneo con UNIX, pero con las características de un *software* libre, y como consecuencia obtuvo GNU (GNU is Not Unix), acrónimo que significa "GNU no es Unix". Con este acontecimiento nació una trayectoria y disputa por el acceso libre a la información, y una libre expresión. Estos esfuerzos se vieron traducidos en la creación en 1985 de la Free Software Foundation (FSF o Fundación para el Software Libre) con el propósito de brindar un soporte logístico, así como legal y financiero al proyecto GNU. La FSF usó programadores para contribuir a GNU; no obstante, la gran parte del desarrollo ha sido causa del trabajo continuo de los voluntarios. A medida que GNU ganaba renombre, negocios, muchos interesados comenzaron a contribuir

[10] Stallman, Richard, *El manifiesto de GNU*, 1983, disponible en: *http://www.gnu.org/gnu/manifesto.es.html*.

al desarrollo o comercialización de productos GNU y el correspondiente soporte técnico.

En esa época, Stallman y un conjunto de desarrolladores que colaboraban con él implantaron dos instrumentos fundamentales para un sistema operativo estilo UNIX: el compilador "GCC" para lenguaje C y el editor de texto "EMACS" (editor de texto) con "Lips" (Interferencias Lógicas por segundo) para cifrar comandos de edición.

Con este acontecimiento, Stallman expidió una licencia que consintiera que los usuarios continuaran usando, copiando, estudiando, modificando o redistribuyendo el *software* instaurado y desarrollado por la FSF, pero que les impidiera adecuarse a las modificaciones que en el futuro ellos mismos ejecutaran, o combinaran el *software* GNU con otro tipo de *software*.

Otro de los hechos relevantes fue el de Roy Tennant, quien en 2007 publicó "el manifiesto del software para bibliotecas", que tiene como finalidad ofrecer un intento por razonar cuál es la relación entre las bibliotecas y los vendedores de sistemas. Este manifiesto es un apoyo de vital importancia para los bibliotecarios, y maneja un orden y una serie de derechos y compromisos que se deben ejercer por parte de los usuarios del *software*:

Derechos como consumidor:

— Tengo derecho a saber lo que existe ahora y cuál es su potencial funcionalidad futura.
— Tengo derecho a usar lo que compro.
— Tengo derecho al API (Application Programming Interface) si he comprado el producto.
— Tengo derecho a documentación completa y actualizada.
— Tengo derecho a mis datos.
— Tengo derecho a tener acceso de sólo lectura a la base de datos.
— Tengo derecho a no hacer las cosas sencillas innecesariamente complicadas.
— Tengo derecho a conocer las líneas de desarrollo y la estimación de tiempo de desarrollo del producto que he comprado.
— Tengo derecho a hacer preguntas técnicas a un equipo capaz de comprenderlas y responderlas.
— Tengo derecho a no ser un probador involuntario.
— Tengo derecho a que se conserven mis personalizaciones y configuraciones en futuras actualizaciones.
— Responsabilidades como consumidor:
— Tengo la responsabilidad de conocer las necesidades de mis usuarios.

— Tengo la responsabilidad de poner las necesidades de mis usuarios por delante de las mías.
— Tengo la responsabilidad de comunicar mis necesidades clara y específicamente.

a. Tengo la responsabilidad de que las mejoras que pido sean realmente lo que quiero.
b. Tengo la responsabilidad de asignar honestamente las prioridades de las mejoras.
c. Tengo la responsabilidad de darme cuenta de que no soy especial.
d. Tengo la responsabilidad de elegir *software* usando un procedimiento limpio y razonable.
e. Tengo la responsabilidad de informar de los errores reproducibles de forma que puedan reproducirse.Tengo la responsabilidad de informar de los errores irreproducibles con todos los detalles que pueda.
f. Tengo la responsabilidad de ver críticamente cualquier ajuste a las configuraciones predefinidas.

Responsabilidades compartidas:

a. Tenemos la responsabilidad de comenzar desde una posición de respeto mutuo.
b. Tenemos la responsabilidad de comunicarnos correctamente.
c. Tenemos la responsabilidad de establecer y mantener un proceso de mejora racional.
d. Tenemos la responsabilidad de mantener las necesidades del usuario final como primordiales.
e. Tenemos la responsabilidad de relajarnos y divertirnos.[11]

Con estos puntos se expresa que cualquier usuario de *software* tiene que cumplir mínimamente con lo expuesto anteriormente.

5. *Características*

El *software* libre tiene sus bases en una ideología, que dice que el *software* no debe tener dueños: es un asunto de libertad. La gente debería ser libre de usarlo en todas las formas que sean socialmente útiles.

[11] Tenant, Roy, *Library Software Manifiesto*, 2007, disponible en: *http://techessence.info/manifiesto/*.

De esta manera, el movimiento del *software* libre pone lo que es beneficioso para la sociedad por encima de los intereses económicos o políticos.

Entre los beneficios que percibe la sociedad se pueden mencionar los siguientes:

1. Tecnologías transparentes, confiables y seguras.
2. Tecnologías como bien público.
3. Promoción del espíritu cooperativo, en el que el principal objetivo es ayudar a su vecino.
4. Precios justos.

El *software* libre ofrece a las personas la posibilidad de utilizar, estudiar, modificar, copiar y redistribuir el *software* (como se ha mencionado de manera recurrente a lo largo del capítulo). Pero para hacer efectivas estas libertades, el código fuente de los programas debe estar disponible.

Gracias a estas libertades se obtienen muchos beneficios prácticos, como los siguientes:

a. Podemos ejecutar el *software* cuando queramos y para lo que queramos.
b. Podemos aprender de los programas existentes.
c. Podemos mejorarlos.
d. Podemos adaptarlos para que se ajusten a nuestras necesidades.
e. Podemos basarnos en ellos, de forma que evitemos los costos adicionales de empezar un programa desde 0.
f. Podemos formar negocios alrededor de la creación, distribución, soporte y capacitación de programas libres.

Existen diferentes aplicaciones del *software* libre entre ellos destacan algunos más usuales, que son:

1. El sistema operativo Linux.
2. El servidor de Web Apache.
3. El manejador de bases de datos objeto-relacional PostgreSQL.
4. El navegador Mozilla.
5. La suite de aplicaciones de escritorio OpenOffice.
6. El servidor de correo Sendmail.[12]

[12] Arriola Navarrete, Óscar y Butrón Yáñez, Katya, "Sistemas integrales para la automatización de bibliotecas basados en *software* libre", *ACIMED*, vol. 18, núm. 6, diciembre de 2008, p. 145.

A. *Ventajas y desventajas*

a. *Ventajas*

En países como Estados Unidos, se le está dando cierto auge al *software* libre, como se puede observar en la revista *Library Journal* de abril de 2010 (Breeding. 2010), ya que cada vez está siendo utilizado por más bibliotecas. Usar *software* libre tiene varias ventajas. Montserrat Culebro menciona las siguientes:

Bajo costo de adquisición y libre uso. El *software*, como mercadería, por lo general no está a la venta. Lo que el usuario adquiere, a través de una erogación monetaria o sin ella, es una licencia respecto de los usos que puede dar a los programas en cuestión. El usuario que adquiere *software* libre lo hace sin ninguna erogación monetaria o a muy bajo costo y ofrece un conjunto de recursos muy amplios. Cualquier persona con una computadora y una conexión a Internet puede utilizar un *software* libre. Para la mayoría de usuarios individuales el *software* libre es una opción atractiva por las libertades que garantiza sin necesidad de verse agobiados por el precio.

Innovación tecnológica. El *software* libre tiene como objetivo principal compartir la información, trabajando de manera cooperativa. Este es principalmente el modelo sobre el que la humanidad ha innovado y avanzado. La ideología de los defensores del *software* libre es que el conocimiento le pertenece a la humanidad, sin hacer distingos. Por lo tanto, los usuarios tienen un destacado papel al influir decisivamente en la dirección hacia donde evolucionan los programas: votando los errores que quieren que sean corregidos, proponiendo nueva funcionalidad al programa, o contribuyendo ellos mismos en el desarrollo del *software* (a finales de 2004 se publicó una lista de las innovaciones más importantes en *software* de 2004. Se consideró como innovación número uno el navegador libre FireFox, y de los diez programas mencionados también se encontraba OpenOffice.org).

Requisitos de hardware *menores y durabilidad de las soluciones*. Aunque resulta imposible generalizar, existen casos documentados que demuestran que las soluciones de *software* libre tienen unos requisitos de *hardware* menores, y por lo tanto, son más baratas de implementar. Por ejemplo, los sistemas Linux que actúan de servidores pueden ser utilizados sin la interfaz gráfica, con la consecuente reducción de requisitos de *hardware* necesarios.

También es importante destacar que en el *software* propietario el autor puede decidir en un momento dado no continuar el proyecto para una cierta plataforma, para un *hardware* que considera antiguo, o descontinuar el

soporte para una versión de su *software*. En las aplicaciones de *software* libre, estas decisiones no pueden ser tomadas por una empresa o individuo, sino por toda una comunidad, con diferentes intereses, lo que se traduce en un mejor soporte de manera general para las versiones antiguas de *software* y de plataformas de *hardware* o *software* minoritarias.

Escrutinio público. El modelo de desarrollo de software libre sigue un método a través del cual trabajan de forma cooperativa los programadores, que en gran parte son voluntarios y trabajan coordinadamente en Internet. Lógicamente, el código fuente del programa está a la vista de todo el mundo, y son frecuentes los casos en que se reportan errores que alguien ha descubierto leyendo o trabajando con ese código.

El proceso de revisión pública al que está sometido el desarrollo del *software* libre imprime un gran dinamismo al proceso de corrección de errores. Los usuarios del programa de todo del mundo, gracias a que disponen del código fuente de dicho programa, pueden detectar sus posibles errores, corregirlos y contribuir a su desarrollo con sus mejoras. Son comunes los casos en que un error de seguridad en Linux se hace público, y con él la solución al mismo. Con el *software* propietario la solución de los errores no llega hasta que el fabricante del programa puede asignar los recursos necesarios para solventar el problema y publicar la solución.

Independencia del proveedor. El *software* libre garantiza una independencia con respecto al proveedor, gracias a la disponibilidad del código fuente. Cualquier empresa o profesional con los conocimientos adecuados puede seguir ofreciendo desarrollo o servicios para nuestra aplicación. En el mundo del *software* propietario sólo el desarrollador de la aplicación puede ofrecer todos los servicios; con el *software* libre, como su denominación lo indica, su uso es libre: todo aquel que lo tiene en su poder puede usarlo cuantas veces quiera, en cuantas máquinas quiera, para los fines que quiera. De esta manera, utilizándolo, el usuario se libera de toda dependencia de un proveedor único, y puede administrar su crecimiento y operación con total autonomía, sin temor de costos ocultos ni extorsiones. Uno de los grandes problemas en la industria del *software* propietario es la dependencia que se crea entre el fabricante y el cliente.

Industria local. Si bien es cierto que no existen aún soluciones libres para todas las necesidades de los usuarios, tampoco existen soluciones propietarias para todas las necesidades. En aquellos casos en que la solución libre no existe, hay que desarrollarla, lo que significa esperar a que alguien más tropiece con la necesidad y lo desarrolle, o desarrollarlo uno mismo (o lo que es igual, pagar para que alguien lo desarrolle). La diferencia está en

que en aquellos casos en que sí hay una solución libre disponible, el usuario puede utilizarla inmediatamente y sin reparos de ningún tipo, mientras que con las soluciones propietarias siempre tiene que pagar, y lo que obtiene a cambio es una "solución" cerrada y secreta, en vez de una herramienta que le permita crecer y operar con seguridad y libertad.

En México, es casi nula la industria de *software*, y las aplicaciones de consumo masivo se desarrollan en otros países. Un gran porcentaje de la industria se basa en distribuir y dar apoyo e información de productos realizados fuera de nuestras fronteras; por lo tanto, la parte de creación y desarrollo de software es realmente la parte de la industria que requiere de excelentes ingenieros y programadores, que sin duda los hay en México, lo que generaría que nuestra industria local creciera generando valor y conocimiento y transcender tecnológicamente.

Datos personales, privacidad y seguridad. Seguridad nacional. Para cumplir con sus funciones, el Estado debe almacenar y procesar información relativa a los ciudadanos. La relación entre el individuo y el Estado depende de la privacidad e integridad de estos datos, que por consiguiente deben ser adecuadamente resguardados contra tres riesgos específicos:

Riesgo de filtración. Los datos confidenciales deben ser tratados de tal manera que el acceso a ellos sea posible exclusivamente para las personas e instituciones autorizadas.

Riesgo de imposibilidad de acceso. Los datos deben ser almacenados de tal forma que el acceso a ellos por parte de las personas e instituciones autorizadas esté garantizado toda la vida útil de la información.

Riesgo de manipulación. La modificación de los datos debe estar restringida, nuevamente, a las personas e instituciones autorizadas.

Adaptación del software. El *software* propietario habitualmente se vende en forma de paquete estándar, que muchas veces no se adapta a las necesidades específicas de empresas y administraciones. Una gran parte de la industria del *software* se basa en desarrollar proyectos donde se requiere *software* personalizado. El *software* libre permite personalizar, gracias al hecho de que se dispone del código fuente, los programas tanto como sea necesario hasta que cubran exactamente las necesidades. La personalización es un área muy importante en que el *software* libre puede responder mucho mejor que el *software* de propiedad a unos costos más razonables. Un gran porcentaje de uso de *software* en los países es de uso interno para empresas y las dependencias de la administración pública, que requiere un alto grado de personalización, y donde el software puede proporcionar desarrollos más económicos.

Lenguas minoritarias, traducción, uso e impulso de difusión. Las lenguas minoritarias existentes en México, como el náhuatl, el zapoteco, el mixteco, el mazahua, el purépecha, entre otros, de nuestras comunidades indígenas, tienen pocas posibilidades de desarrollarse en el mundo del *software* propietario y para aquellas personas que no dominan el castellano, y sólo la lengua original de la comunidad no tendría acceso al uso y manejo de las computadoras, además de que iría perdiendo cada vez más estos idiomas, aunque muchos quizá opinen que este no es una ventaja importante o un tema relevante; pero pensamos que podría servir como un medio para impulsar la difusión de estas lenguas y que no queden en el olvido y se pierda parte de esta cultura, y por lo tanto de nuestro patrimonio nacional.

b. *Desventajas*

El uso adecuado del *software* libre puede representar grandes ahorros y beneficios a una biblioteca, aunque también es relevante marcar ciertas desventajas que puede representar, como:[13]

El aprendizaje de los usuarios es menor. Si se pone a dos señoras que nunca han tocado una computadora, probablemente tardarán lo mismo en aprender a usar *software* propietario; por ejemplo, de Microsoft, que *software* libre, como Gnome o KDE; pero si antes los usuarios ya usaron *software* propietario, generalmente tardan más en aprender a usar un *software* libre.

El software *libre no tiene garantía proveniente del autor*. Se puede hacer uso libre del paquete, e incluso modificarlo, pero el autor no se hace responsable por cualquier tipo de falla o inconveniente que pueda surgir.

Los contratos de software *propietario no se hacen responsables por daños económicos, y de otros tipos por el uso de sus programas.* El *software* libre se adquiere, se vende "AS IS" (tal cual) sin garantías explicitas del fabricante; sin embargo, puede haber garantías específicas para situaciones muy específicas.

Se necesita dedicar recursos a la reparación de errores. Sin embargo, en el *software* propietario es imposible reparar errores, hay que esperar a que saquen a la venta otra versión.

No existen compañías únicas que respalden toda la tecnología. Para el desarrollo de un *software* se hacen trabajos cooperativos, donde muchas personas están involucradas; este grupo de personas pueden tener cambios constantes de personal, incluso los usuarios u otras organizaciones pueden aportar algo al desarrollo del *software*; por esta razón, no hay una compañía u organización única que respalden por completo el *software*.

[13] Culebro Juárez…, *op. cit.*

Las interfaces gráficas de usuario (GUI) y la multimedia apenas se están estabilizando. Aunque hay un número cada vez mayor de usuarios que aseguran que las interfaces gráficas más populares en el *software* libre (KDE, GNOME y el manejador de ventanas WindowMaker) son ya lo suficientemente estables para el uso cotidiano y lo suficientemente amigables para los neófitos de la informática.

La mayoría de la configuración de hardware *no es intuitiva*. Se requieren conocimientos previos acerca del funcionamiento del sistema operativo y fundamentos del equipo a conectar para lograr un funcionamiento adecuado. Sin embargo, la documentación referente a la configuración del *hardware* es tan explícita y detallada que permite al usuario neófito, profundizar en el conocimiento de su *hardware* en muy pocas horas, y una vez teniendo ese conocimiento la configuración se vuelve trivial.

Únicamente los proyectos importantes y de trayectoria tienen buen soporte, tanto de los desarrolladores como de los usuarios. Sin embargo, existen muchos proyectos más pequeños y recientes que carecen del compromiso necesario por parte de sus usuarios o desarrolladores para que sean implementados de manera confiable. Estos proyectos importantes que tienen un excelente soporte cubren más del 90% de las necesidades de cómputo del usuario promedio.

El usuario debe tener nociones de programación. La administración del sistema recae mucho en la automatización de tareas, y esto se logra utilizando, en muchas ocasiones, lenguajes de guiones (*perl*, *python*, *shell*, etcétera). Sin embargo, existen en la actualidad muchas herramientas visuales que permiten al usuario no técnico llevar a cabo tareas de configuración del sistema de una manera gráfica muy sencilla sin la necesidad de conocimientos de programación.

En sistemas con acceso a Internet se deben monitorear constantemente las correcciones de errores de todos los programas que contengan dichos sistemas, ya que son fuentes potenciales de intrusión. En el *software* propietario también se deben monitorear constantemente las correcciones de errores de todos los programas, y además es imposible reparar las vulnerabilidades (que en su mayoría son reparaciones triviales) por uno mismo, sino que hay que esperar a que la compañía fabricante libere la actualización, y en algunos casos hay que pagar dinero extra por obtener esta.

La diversidad de distribuciones, métodos de empaquetamiento, licencias de uso, herramientas con un mismo fin, etcétera, pueden crear confusión en cierto número de personas. Hay quienes ven esto como una fortaleza porque se pueden encontrar desde distribuciones especializadas en sistemas embebidos con muchas limitantes de almacenamiento y dispositivos periféricos de uso especializado hasta dis-

tribuciones optimizadas para su uso en servidores de alto rendimiento con varios procesadores y gran capacidad de almacenamiento, pasando por las distribuciones diseñadas para su uso en computadoras de escritorio, y entre las cuales se encuentran las diseñadas para el usuario neófito, que son muy fáciles de instalar y utilizar, y las diseñadas para el usuario avanzado con todas las herramientas necesarias para explotar el *software* libre en todo su potencial. Cabe notar que la posibilidad de crear distribuciones completamente a la medida para atacar situaciones muy específicas es una ventaja que muy pocas marcas de *software* propietario pueden ofrecer, y que Microsoft ha sido completamente incapaz de hacer.

VI. Reflexiones finales

Con las nuevas tendencias en cuanto a normatividad para la organización de la información, acceso a los recursos electrónicos y la gestión de colecciones digitales, la oferta actual de sistemas de automatización se ha ido convirtiendo en un obstáculo para el progreso. Está emergiendo una nueva generación de plataformas de servicios digitales para bibliotecas, diseñadas para proporcionar un apoyo integral a la gestión y al acceso de todo tipo de materiales de la biblioteca: impresos, electrónicos y digitales. Estos nuevos sistemas implicarán una modernización de las arquitecturas orientadas a servicios, con un mayor desarrollo del concepto de "*software* como servicio" y de otros modelos basados en la "nube".

Las plataformas completamente basadas en la Web implican un modo de colaboración intensiva entre los grupos de bibliotecas para aumentar el impacto de sus colecciones, reducir los costos de las operaciones, dedicar menos recursos en la infraestructura de tecnología local, y agilizar los procesos relacionados con la gestión de sus colecciones, consolidándose como la mejor arquitectura capaz de soportar estas estrategias.[14]

Los grandes desarrolladores de *software* para bibliotecas se han dado cuenta, tras un período de investigación, que deben ofrecer opciones alternativas a las bibliotecas, tanto en la automatización del *back-end* (operaciones internas de los bibliotecarios) como en el descubrimiento (búsqueda y localización de información por parte del usuario final). Así, han surgido varias soluciones nuevas, que a menudo representan modelos conceptuales muy diferentes, que van más allá del formato MARC21; los bibliotecarios siempre buscan hallar soluciones que mejoren las experiencias de sus usuarios de forma inmediata, especialmente a través de productos de descubrimiento.

[14] Breeding, Marshall, "Library Systems…", *cit.*, pp. 30-43.

Este nuevo *software* de automatización surgirá en el contexto de hechos y circunstancias que moldean el trabajo de las bibliotecas universitarias, especialmente en relación con la proporción cada vez mayor de materiales electrónicos y digitales que componen las colecciones.

El modelo tradicional de SIAB no es muy adecuado para la gestión de recursos electrónicos, y ello ha llevado a la proliferación de productos complementarios tanto de *software* libre como de *software* propietario, para abordar este aspecto cada vez más estratégico para el funcionamiento de las bibliotecas, en donde además de su SIAB implementan servicios basados en Open URL para la resolución de enlaces, sistemas especializados de gestión de recursos electrónicos, interfaces de descubrimiento, plataformas de gestión de activos digitales (*digital asset management*), repositorios institucionales, servidores *proxy* y otros componentes que abordan un aspecto u otro de las operaciones digitales.

Cada una de estas aplicaciones suele requerir la implementación de procesos de gestión y plataformas de *hardware* separadas, de lo cual resulta un entorno muy complejo de gestión, que supera la capacidad de muchas bibliotecas. Estos sistemas a menudo no logran interactuar entre sí con efectividad, debido a que usan modelos de datos aislados y a la falta de API robustas y bien desarrolladas.

Para 2016 se esperaba la implementación de plataformas de servicios bibliotecarios más completas y adecuadas. Hay que tener en cuenta que actualmente muchas bibliotecas todavía usan SIAB diseñados hace más de una década; esto en el contexto internacional; en el ámbito mexicano, el tema es más grave, porque estamos hablando en su mayoría de bibliotecas con recursos limitados, las que no tienen más remedio que tratar de sacar el mejor partido posible de su sistema de automatización obsoleto. Históricamente, la plena realización del ciclo de vida de los productos es de una década.

Breeding comenta que mirando el calendario de 2026 se puede aventurar que las plataformas de servicios bibliotecarios habrán alcanzado la plena madurez, y su despliegue será casi universal. De la misma manera que casi cualquier biblioteca universitaria de hoy ha puesto en marcha un sistema integrado de biblioteca de algún tipo, para 2026 se puede esperar que sea usual disponer de plataformas capaces de manejar exhaustivamente recursos impresos, electrónicos y digitales.

Muchos *softwares* de automatización utilizados en las bibliotecas universitarias actuales se habrán extinguido. De todas formas, es de esperar que también sobrevivan algunos de ellos, pero la década de evolución los hará casi irreconocibles con respecto a su forma actual, puesto que habrán ad-

quirido muchas de las características de las plataformas de servicios bibliotecarios puestas en marcha en 2012. El entorno de automatización de bibliotecas ha sido siempre favorable a un enfoque evolutivo para el desarrollo de productos, a pesar de que el ciclo actual presenta una serie de alternativas revolucionarias. Algunos proveedores han sido capaces de navegar con éxito a través de una larga serie de ciclos tecnológicos, como Innovative Interfaces, Inc., VTLS, SirsiDynix (y sus compañías predecesoras) y Ex Libris. Podemos anticipar que éstos, y otros, seguirán haciendo evolucionar los productos existentes o crearán otros nuevos a lo largo de la próxima era de la automatización de bibliotecas.[15]

VII. Bibliografía

Arriola Navarrete, Óscar, "Open access *y* software libre: un área de oportunidad para las bibliotecas", *Biblioteca Universitaria. Revista de la Dirección General de Bibliotecas de la UNAM*, vol. 14, núm. 1, enero-junio de 2011.

Arriola Navarrete, Óscar y Butrón Yáñez, Katya, "Sistemas integrales para la automatización de bibliotecas basados en software libre", *ACIMED*, vol. 18, núm. 6, diciembre de 2008.

Breeding, Marshall, "Automation System Marketplace 2010: New Models, Core Systems. Discovery Interfaces Add a New Facet to the Marketplace", *Library Journal*, vol. 135, núm. 6, 1 de abril de 2010.

Breeding, Marshall, "Current and Future Trends in Information Technologies for Information Units", *El Profesional de la Información,* vol. 21, núm. 1, 2011.

Breeding, Marshall, "Library Systems Report 2016: Power plays", *American Libraries: the Magazine of the American Libraries Association*, vol. 47, núm. 5, mayo de 2016.

Carranza Torres, Martín, *Problemática jurídica del software libre,* Buenos Aires, Lexis-Nexis, 2004.

Culebro Juárez, Montserrat, *Software libre vs. Software propietario: ventajas y desventajas*, 2006, disponible en: *http://www.softwarelibre.cl/drupal//files/32693.pdf*.

Da Rosa, Fernando y Heinz, Federico, *Guía práctica sobre software libre su selección y aplicación local en América Latina y el Caribe*, Organización de las Nacio-

[15] Breeding, Marshall, "Current and Future Trends in Information Technologies for Information Units", *El Profesional de la Información,* vol. 21, núm. 1, 2011, pp. 9-15.

nes Unidas para la Educación la Ciencia y la Cultura, 2007, disponible en: *http://unesdoc.unesco.org/images/0015/001560/156096s.pdf*.

DAVID, Lourdes T., "Introduction to Integrated Library Systems", *ICT for Library and Information Professionals: A Training Package for Developing Countries*, 2001, disponible en: *http://arizona.openrepository.com/arizona/bitstream/10150/106374/1/125105e.pdf*.

GNU Operating System, Free Software Foundation, 2009, disponible en: *http://www.gnu.org/philosophy/free-sw.html*.

GÓMEZ SÁNCHEZ, Rafael, "Software libre vs. Software propietario: programando nuestro futuro", *HAOL*, núm. 2, otoño de 2004.

"Open source software", *Postnote*, núm. 242, junio de 2005, disponible en: *http://www.parliament.uk/documents/post/postpn242.pdf*.

PERENS, Bruce, *Open Standards, Principles and Practice*, disponible en: *http://perens.com/OpenStandards/Definition.html*.

PORCEL ITURRALDE, María Laura y RODRÍGUEZ MEDEROS, Mabel, "Software libre: una alternativa para las bibliotecas", *ACIMED,* vol. 13, núm. 6, 2005.

RODRÍGUEZ, Gladys Stella, "El software libre y sus implicaciones jurídicas", *Revista de Derecho*, núm. 30, 2008.

SAMPEDRO, José Luis, "Construcción de capacidades de innovación en la industria de software a través de la creación de interfaces: el caso de empresas mexicanas", *1er Congreso Iberoamericano de Ciencia, Tecnología, Sociedad e Innovaciones CTS+I,* 2006.

STALLMAN, Richard, *El manifiesto de GNU*, 1983, disponible en: *http://www.gnu.org/gnu/manifesto.es.html*.

STALLMAN, Richard, *Software libre para una sociedad libre,* 2002, disponible en: *http://biblioweb.sindominio.net/pensamiento/softlibre/softlibre.pdf*.

TENNANT, Ro, *Library software manifiesto,* 2007, disponible en: *http://techessence.info/manifesto/*.

WAYNER, Peter, *La ofensiva del software libre. Cómo Linux y el movimiento del software libre se impusieron a los titanes de la alta tecnología*, Barcelona, Garnica, 2001.