Reflexiones en torno a la arqueología de medios en el software libre

Resumen

Con la construcción de una arqueología de medios del software libre, este texto busca reflexionar en torno a las razones de su surgimiento y a las amenazas que representa para los creadores el no tener control sobre las tecnologías que se usan en el contexto de sus proyectos de creación. Además, examina una de las principales características que se supone tiene el software libre: se puede modificar el software que se usa, estudiarlo y hacerle cambios al código fuente, aunque en la práctica no todas las personas cuentan con las competencias necesarias para hacerlo. Este texto se interesa por las posibles oportunidades creativas que pueden surgir cuando diseñadores, artistas y creadores dejan de ser consumidores pasivos de tecnologías para el ejercicio de su quehacer profesional. Asimismo, analiza el problema de que solo se adopten tecnologías y la importancia de que se pueda tener más control sobre ellas; especialmente para adaptarlas a las necesidades específicas en los procesos de creación. Este texto presenta una postura crítica sobre varios aspectos de la filosofía y el uso práctico del software libre en el contexto de la creación.

José David Cuartas Correa
Doctor en Diseño y Creación
Director Laboratorio Hipermedia
Fundación Universitaria
Los Libertadores, Bogotá, Colombia
Correo electrónico:
jdcuartac@libertadores.edu.co
orcid.org/0000-0001-7211-2237
Google Scholar

Recibido: Abril 7 de 2017
Aprobado: Agosto 10 de 2017

Palabras clave:
Arqueología de medios, código fuente, oportunidades creativas, programación para no programadores, software libre.

Revista KEPES Año 15 No. 17 enero-junio 2018, págs. 69-92 ISSN: 1794-7111(Impreso) ISSN: 2462-8115 (En línea)
DOI: 10.17151/kepes.2018.15.17.4

Revista KEPES, Año 15 No. 17, enero-junio de 2018, págs 69-92

Reflections on media archeology free software

Abstract

With the construction of media archeology free software, this text seeks to reflect on the reasons for its emergence and the threats it represents for creators not having control over the technologies used in the context of their creation projects. In addition, it examines one of the main characteristics that free software is supposed to have: the software used can be modified, can be studied and changes to the source code can be made, although, in practice, not everyone has the necessary skills to do it. This text is interested in the possible creative opportunities that can arise when designers, artists and creators stop being passive consumers of technologies for the exercise of their professional work. It also analyzes the problem of only adopting technologies and the importance of having more control over them, especially to adapt them to the specific needs in the creation processes. This text presents a critical stance on various aspects of the philosophy and the practical use of free software in the context of creation.

Key words:
Archeology of media, source code, creative opportunities, programming for non-programmers, free software.

Introducción

Este artículo recurre a la construcción de una arqueología de medios del software libre con la cual se "hurga archivos de texto, visuales, auditivos, así como colecciones de artefactos" (Huhtamo and Parikka, 2011, p. 3) como insumos para reflexionar en torno a las razones del origen de este movimiento. Se elige aplicar la arqueología de medios, ya que "en una perspectiva pragmática significa excavar caminos secretos en la historia, que nos pueden ayudar a encontrar el camino hacia un futuro" (Zielinski, 1996, p. 11); de forma tal que se identifiquen las amenazas que puede llegar a representar para los creadores no tener el control sobre las tecnologías que se usan en el contexto de la creación y el diseño.

Este artículo está construido haciendo uso de dos tipos de instrumentos metodológicos diferentes, pero complementarios. Uno de ellos es una arqueología de medios, que se usa como una estrategia para contextualizar al lector en torno a las temáticas de las tecnologías libres y las culturas *hacker* y *maker*. El segundo instrumento es un análisis documental, el cual se usa para abordar el dilema de programar o ser programados para despertar el interés acerca de la desobediencia tecnológica y la mal utilización de las tecnologías. Los criterios de selección de fuentes se basaron en rastrear, a mayor profundidad, algunas de las menciones superficiales hechas por expertos tales como Victor (2013), Elliott (2008), Brand (1988) y Levy (2010).

Para iniciar esta arqueología es importante revisar lo que aconteció en los primeros días de la historia de la computación. En las primeras décadas de la computación, en su gran mayoría, el software era desarrollado por los mismos usuarios y era compartido de forma usual entre la comunidad de académicos e investigadores. El software era un recurso que circulaba libremente debido a que no se podía ejecutar un programa escrito para una máquina Data General

en una máquina IBM, por lo que Data General e IBM no se preocuparon mucho por controlar su software (Lessig, 2004). A finales de la década de los 60, las computadoras se hicieron cada vez más potentes; al comprarlas estas venían con un software precargado, el cual estaba incluido en el costo de venta de cada una. Como lo explica Lessig (2004), en ese entonces, cuando las computadoras con software se pusieron a disposición comercialmente, el software —tanto el código fuente como los binarios— era gratuito.

En la década de los 70, aparecen las primeras computadoras personales (entre ellas Altair y Apple I) gracias a que el hardware disponible se hizo cada vez más potente y menos costoso (esto como consecuencia de la ley de Moore). Sin embargo, al final de esta década, cuando Bill Gates y Paul Allen:

> inventaron la idea de vender software, [pero] se encontraron con la crítica tanto de los hackers como de los sobrios hombres de negocios. Los hackers entendían que el software sólo era información, y le ponían objeciones a la idea de venderla. Estas objeciones eran en parte morales. Los hackers salían del mundo científico y académico, donde resulta imperativo hacer que los resultados del propio trabajo queden disponibles para el público. También eran en parte objeciones prácticas: ¿cómo puedes vender algo que puede copiarse fácilmente? (Stephenson, 2003, p. 38)

Así que el software se convierte en terreno de batalla y se le declara la guerra a la práctica de compartirlo; principalmente motivada por la carta abierta a los usuarios por *hobby* escrita por Bill Gates en 1976. En esta carta, Gates (1976) manifestaba: "la mayoría de vosotros roba el software. El hardware se tiene que pagar, pero el software es algo que se comparte" (p. 4). Una acusación muy agresiva, a una práctica heredada de los primeros días de la computación. Por su parte, Gates afirmaba que "lo que se consigue es impedir que se fabrique buen software" (p. 5). Esto en el modelo de innovación cerrada de aquel entonces tenía sentido, pero hoy con la innovación abierta se ha comprobado que se puede producir software de muy buena calidad mediante la cooperación de desarrolladores de todo el mundo. Entre algunos ejemplos

que se pueden mencionar está el sistema operativo *GNU/Linux*, el software para edición de imágenes vectoriales *Inkscape*, el editor para imágenes de mapas de bits *Gimp* o el lenguaje de programación para artistas y diseñadores *Processing*.

Gates (1976), también se preguntaba: "¿qué usuario por hobby puede dedicar el trabajo de tres hombres durante tres años buscando bugs, documentando el producto, distribuyéndolo de manera gratuita?" (p. 5); lo cual ya ha sido demostrado por la gran cantidad de desarrolladores que contribuyeron de forma incondicional a proyectos de software libre tales como Ubuntu, Android, Firefox, Blender, Audacity, Processing y muchos otros. Gates (1976), por último, también se preguntaba: "¿quién se puede permitir el hacer una labor profesional a cambio de nada?" (p. 5). Esta afirmación puede interpretarse de dos formas como si fueran las dos caras de una misma moneda. Por un lado Gates tiene razón debido a que solo algunas figuras como Richard Stallman (creador del movimiento del software libre) no necesitan preocuparse por sus finanzas, pues son pioneros y viven de su prestigio adquirido (fama). No obstante, los mayores entusiastas, y quienes más contribuciones hacen al movimiento de software libre, usualmente son jóvenes universitarios de pregrado que tampoco tienen que preocuparse por sus finanzas puesto que son sus padres quienes los mantienen o en otros casos viven de sus becas de estudio (González, Seoane y Robles, 2008).

Por otro (la otra cara de la moneda), Gates se equivocaba debido a que en su mente no estaba la opción de que las personas pudieran producir software por el mismo placer de hacerlo y por el deseo de pertenecer a una comunidad donde pudieran ser reconocidos por sus aportes. Básicamente es una cuestión de reputación y prestigio intelectual, que va más allá de recompensas económicas. Fenómeno definido por Chris Anderson como la economía del regalo (*Gift economy*) que se fundamenta en dar algo a cambio de recompensas que no son

necesariamente monetarias, sino que su principal motivación es la de mejorar la reputación o el estatus dentro de un grupo o comunidad (Anderson, 2009).

Con esta carta se declaraba la guerra. Pero con lo que no contaba Gates era que unos años después en el corazón del Instituto Tecnológico de Massachusetts —MIT—, particularmente en el laboratorio de inteligencia artificial, un *hacker* —Richard Stallman— iba a abanderar un esfuerzo sin precedentes para construir un sistema operativo totalmente libre que denominó GNU. Nombre que proviene del acrónimo recursivo: *GNU's Not Unix!*

Esto debido a que el diseño de GNU es similar y compatible con Unix (sistema operativo creado en 1969); aunque es muy diferente, ya que no usa su código y es 100 % software libre. Así pues, en 1983, Stallman anuncia en la USENET (una red de grupos de noticias que surge en 1980) la creación del proyecto GNU (Stallman, 1983). En 1985, publica el manifiesto GNU que fundamentará las bases filosóficas e ideológicas del movimiento de software libre; impulsado desde la Free Software Foundation fundada también por Stallman.

Se podría decir que la iniciativa de Stallman obedece a lo que Victor define como *fight by inventing* (luchar mediante la invención) y plantea que cualquier experto en tecnología o inventor puede reconocer un error en el mundo; puede tener una visión de lo que podría ser un mundo mejor y puede dedicarse a luchar por un principio. En lo social los activistas suelen luchar usando estrategias de organización (marchas, protestas), pero ellos también pueden luchar inventando (Victor, 2012, 37:48).

Es a partir de 1986 que se comienza a construir la definición de software libre (Stallman, 1986), que a la fecha se condensa en el cumplimiento de un conjunto de cuatro libertades esenciales: libertad 0: ejecutar; libertad 1: estudiar y modificar; libertad 2: copiar; libertad 3: distribuir modificaciones.

Libertad 0: ejecutar.
Libertad de ejecutar el programa como usted lo desee.

Libertad 1: estudiar y modificar.
Libertad de ayudarse usted mismo, esta es la libertad de estudiar el código fuente y cambiarlo para hacer lo que usted desee.

Libertad 2: copiar.
Libertad de ayudar a sus vecinos, esta es la libertad de hacer copias y distribuirlas a otros cuando usted lo desee.

Libertad 3: distribuir modificaciones.
Libertad de ayudar a su comunidad, esta es la libertad de publicar y distribuir una versión modificada cuando usted lo desee.

Fuente: íconos (Hancock, 2008).

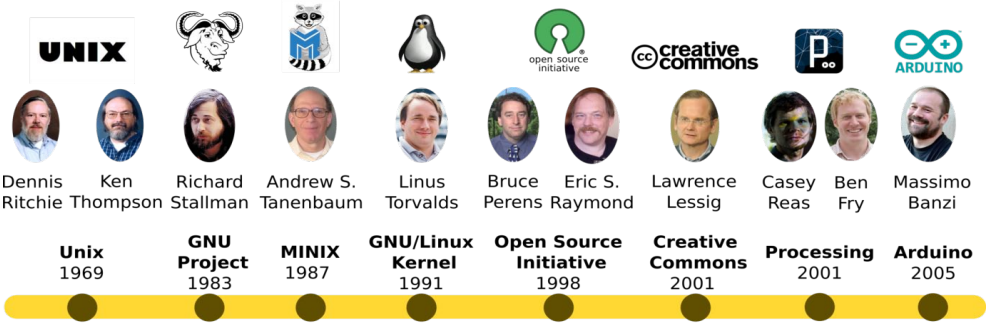
Por otra parte, el término software libre (*free software*) en la lengua inglesa tiene una ambivalencia que hace que "muchas personas piensen erróneamente que significa sólo software de costo cero. Es cierto que todo el software libre es de costo cero, pero no todo el software de coste cero es libre" (Fogel, 2005, p. 16). En este punto cabe agregar un dato curioso y es que en los años 80, según Brand (1988), "la industria del entretenimiento se refería de forma rutinaria, al contenido guardado en los distintos medios de almacenamiento, con el termino *software*, el cual provenía del mundo de las computadoras" (p. 29); y bajo esta definición Heyer (1988), agrega: "la grabación con videocasetera proporcionó una fuente ilimitada de software aparentemente libre" (como se

citó en Brand, 1988, p. 29). Es decir que el termino *free software* se utilizaba en los años 80 para referirse a la circulación de contendidos, siendo Stallman el que crea un nuevo significado para este mismo término.

Entonces, para diferenciar al software libre del software gratis, se debe aclarar que el software libre no debe considerarse como un sinónimo de algo gratuito. Lo menos importante del software libre es que sea gratuito, tanto así que el software libre se puede vender. Lo más importante del software libre es que precisamente ofrece la libertad para poder ejecutar, modificar, copiar y distribuir modificaciones del software.

Después de la creación del proyecto GNU se pueden identificar otros actores clave en el proceso de expansión del movimiento de software libre (gráfica 1). Uno de ellos es el *hacker* Linus Torvalds, quien se une a la causa aportando en el año 1991 uno de los componentes faltantes del proyecto GNU: el *kernel*; el cual se necesitaba para que el sistema funcionara en máquinas de arquitectura Intel en la que se basaban los PC de la época. El *kernel* es una parte del sistema operativo que se encarga de gestionar los recursos para que los programas puedan tener acceso seguro al hardware de la computadora. Este *kernel*, creado por Torvalds, se basaba en un sistema operativo de la época llamado MINIX (que fue desarrollado por Andrew Tanenbaum como ejemplo para sus clases). En 1998, Bruce Perens y Eric Raymond crean el movimiento *Open Source*; movimiento que se preocupa más por los aspectos prácticos de compartir el código, que por los aspectos éticos. Tres años después, en 2001, Lawrence Lessig crea el movimiento *Creative Commons* (equivalente del software libre en el contexto de los contenidos). En el mismo año surge un proyecto de software libre en el Media Lab del MIT llamado *Processing*, liderado por Casey Reas y Ben Fry. Este proyecto le aportó a la comunidad de artistas y diseñadores una herramienta para el rápido prototipado de aplicaciones interactivas. Por último, es importante mencionar el proyecto *Arduino* promovido por Massimo

Banzi; el cual influye significativamente para que se cree una masa crítica en torno al desarrollo de hardware abierto para proyectos de arte y diseño.



Gráfica 1. Historia comúnmente conocida del software libre.
Fuente: elaboración propia por parte del autor.

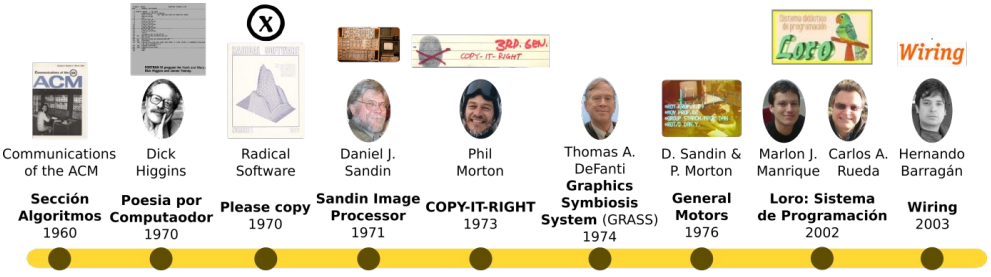
No obstante, existen una serie de historias relativamente desconocidas en el contexto del software libre y los contenidos libres que valen la pena divulgar (gráfica 2). Una de ellas es que desde los años 60, a partir del volumen 3 de la revista *Communications of the ACM* ya se publicaba en la sección 'Algorithms' ejemplos de algoritmos de software para diferentes propósitos. Esto en cierta medida puede considerarse como software libre, ya que se hacía público el código fuente para que otras personas lo usaran y modificaran de acuerdo a sus necesidades.

De la misma forma, en la década de los 70, el artista Dick Higgins —promotor del movimiento *Fluxus*— publica el código fuente del poema "Hank and Mary"; creado haciendo uso del lenguaje de programación *FORTRAN IV*, en su libro *Computers for the arts* (1970).

Otra historia desconocida es que el concepto de contenidos tipo *Creative Commons* ya había sido explorado 30 años antes de que fuera acuñado así, en 2001. Este es el caso de la revista *Radical Software*, que fue publicada entre 1970 y 1974, cuya área de interés era la manipulación de vídeo. Esta revista buscaba demostrar su compromiso con el libre acceso a la información, por lo que rechazaba el estándar de derechos de autor en favor de uno nuevo; un círculo con una X adentro, significando: "Por favor copie" (Gigliotti, 2003).

Por los mismo años, también aparece la licencia *COPY-IT-RIGHT* creada por Morton (1973) para distribuir los circuitos del *Image Processor* de Sandin con el nombre de *Distribution Religion*. Con el *Image Processor*, Morton y Sandin (1976, 15:20) generaban audiovisuales (uno de ellos llamado: General Motors) haciendo uso de graficas generadas por computador que programaban en tiempo real (*live coding*) con el leguaje *GRASS* creado por Tomas DeFanti en el año 1974.

En el contexto colombiano se debe mencionar que, en 2002, paralelo al proyecto *Processing*, se creó un lenguaje llamado Loro para facilitar el aprendizaje de conceptos básicos de programación; licenciado como software libre (Manrique y Rueda, 2002). Otra historia que no es del todo conocida es la del proyecto *Wiring*, creado 2 años antes del proyecto *Arduino* por Hernando Barragán; quien se dio a la tarea de migrar el entorno de programación *Processing* para que pudiera usarse como entorno para programar microcontroladores. Con el propósito de ofrecer una plataforma que fuera fácil de usar por artistas y diseñadores para el prototipado de proyectos de computación física (Barragán, 2003).



Gráfica 2. Historias desconocidas del software libre.
Fuente: elaboración propia por parte del autor.

Otros autores, que también han hecho aportes significativos al software libre y no quedaron en la anterior línea de tiempo, son: Andrew Keith Paul Morton, quien es uno de los desarrolladores líder del *kernel* de *GNU/Linux* y encargado de mantener el sistema de archivos ext3; Andrew Tridgell, quien es el creador del servidor de archivos Samba para *GNU/Linux*; Mark Shuttleworth, quien es el fundador de Canonical y de la distribución *GNU/Linux* llamada Ubuntu; Marc Ewing, quien es el fundador de la distribución *GNU/Linux* llamada Red Hat y Miguel de Icaza, un programador mexicano creador del proyecto *GNOME* (un entorno gráfico para las distribuciones *GNU/Linux*).

Reflexiones

En la libertad 1 se establece que el software libre debe poderse modificar y debe poderse estudiar y hacerle cambios al código fuente, pero en muchos casos es reducido el número de personas con las competencias técnicas necesarias para hacerlo. Incluso el software libre puede terminar siendo tan cerrado como el software privativo dado que, por un lado, compilarlo desde su código fuente puede llegar a ser una tarea muy frustrante sobre todo para los no expertos. Y, por otro, porque muchas veces no se cuenta con los suficientes conocimientos

técnicos para entender la forma como fue escrito el código fuente para poder modificarlo. No obstante, para Stallman, "su objetivo fundamental era la libertad; el código creativo e innovador era un subproducto" (como se citó en Lessig, 2004, p. 280).

Preocuparnos solamente por los aspectos éticos y políticos del software libre pone en ventaja a una élite intelectual de *hackers*, académicos, programadores y *makers*, aunque no necesariamente beneficia de forma directa a los usuarios no expertos.

Muchas tecnologías se implementan con el ideal de potenciar la creatividad y expresividad de los artistas y creadores, pero terminan convirtiéndose en herramientas que dictan lo que se puede hacer o no. Es por ello que no debemos seguir permitiendo que sea la tecnología la que nos diga lo que podemos o no hacer, si es que no queremos caer en un estado de estandarización o estancamiento creativo.

Por otra parte se puede decir que cada promotor de un nuevo avance en la tecnología anhela mejorar el mundo, el problema es que estos anhelos muchas veces son incompatibles con los anhelos de los promotores de otros cambios e innovaciones y muchas veces terminamos siendo "prisioneros de un mundo de fantasía construido de piezas incompatibles provenientes de sueños rotos" (Nelson, 2008, p. 3).

Es importante reflexionar sobre las posibles oportunidades creativas que pueden surgir cuando diseñadores, artistas y creadores dejan de ser consumidores pasivos de tecnologías para el ejercicio de su quehacer profesional. Reflexionar sobre el riesgo de que solo adopten tecnologías, y sobre la importancia de que se pueda tener más control sobre ellas, especialmente para adaptarlas a las necesidades específicas en sus procesos creativos. Así pues, es importante

asumir una postura frente a la pregunta: ¿*adoptar* tecnologías o *adaptar* tecnologías? Y aunque la diferencia entre ambas es de tan solo una letra, cambia enormemente su significado y sentido; ya que mientras *adoptar* es sinónimo de 'dependencia' y de 'centralización' y obedece a la lógica del software propietario. *Adaptar* es sinónimo de 'autonomía' y de 'descentralización' y obedece a la lógica del software libre.

Es clave identificar oportunidades creativas en el uso de tecnologías que empoderen a los creadores. En este sentido Ted Nelson (2008), uno de los pioneros de la informática, bien conocido por acuñar los términos hipertexto e hipermedia, afirma que ve al mundo de la computación en la actualidad como "una prisión de pesadilla, ruidosa y colorida y totalmente bastarda […] somos prisioneros en las aplicaciones que se pueden personalizar sólo en las formas que los diseñadores permiten" (p. 199).

Nelson (2008), en este mismo orden de ideas, define esto como consecuencia de uno de los mitos tecnológicos que hace pensar a las personas que las tecnologías son recursos predeterminados y que no se pueden modificar. "Este mito de la necesidad tecnológica ha sofocado imaginación de la gente. Y su voluntad de exigir más y mejor" (p. 192). Y es que muchas de las cosas que pueden o no hacer las tecnologías, muchas veces solo depende de las decisiones ideológicas y políticas de las personas que le dan forma y no de verdaderas limitaciones técnicas.

Nelson también afirma que otro de los tantos mitos sobre las tecnologías es pensar que el empaque es la tecnología misma, a lo que pone como ejemplo que un iPhone o el software de Microsoft no son tecnologías sino empaques. Y puesto que el envoltorio es lo que ven las personas, esto reafirma la creencia de que estos empaques son tecnologías (Nelson, 2012, 2:33). Y lo peor sucede cuando estos empaques deciden que podemos o no hacer con nuestras

computadoras. Cuando las tecnologías nos permiten hacer cosas imaginables, los empaques limitan nuestro potencial creativo decidiendo lo que ellos creen que es mejor para nosotros.

Otro aspecto sobre el cual se debe reflexionar es si la creatividad se debería limitar a un uso exclusivo de tecnologías abiertas. Si, por ejemplo, una tecnología privativa empodera más al creador entonces los ideales del software libre no deben verse pisoteados si se usa dicha herramienta; de lo contrario, estaría el creador en un lugar de desventaja. Por encima de la filosofía de usar software libre debe estar el empoderamiento del creador. Se debe considerar el software como una capa de las tecnologías que puede empoderar o enajenar. Y es que muchas veces lo que realmente limita las posibilidades creativas son aspectos legales y éticos, más que propiamente limitaciones técnicas.

La importancia de mal utilizar las tecnologías

Desde otro ángulo de reflexión es importante retomar lo que afirma Erkki Kurenniemi, uno de los pioneros en los años 60 de la música electrónica en Finlandia (imagen 1), “la tecnología no tomará el control mientras el hombre pueda mal utilizarla” (como se citó en Taanila, 2003, 12:21).

De esta afirmación surge una pregunta: ¿cómo podemos mal utilizar la tecnología si es que no se puede modificar, adaptar o estudiar? Sobre todo porque un acto altamente creativo, es darle a las tecnologías existentes usos insospechados.

Imagen 1. Erkki Kurenniemi programando (1967).
Fuente: Taanila (2003).

En este mismo orden de ideas, Victor (2013) manifiesta que "siempre he considerado la computadora como un medio artístico" (00:06); pero en la época que Kurenniemi usaba computadoras como un medio artístico para hacer música, esto era considerado por los ingenieros e investigadores en ciencias computacionales como una desperdicio de recursos, de tiempo y de dinero (prácticamente, un sacrilegio).

No obstante es gracias a este acto de mal utilizar las tecnologías, para encontrar usos insospechados, que podemos descubrir el verdadero potencial de las computadoras; ya que, como lo expresa Victor (2013), "debemos tener en cuenta que este sigue siendo un medio muy joven y todavía estamos muy lejos de entender el verdadero potencial de este medio" (00:12).

Se ha creído utópicamente que los nuevos medios y las tecnologías emergentes potencian los procesos creativos, pero cómo lo manifiesta Machado (2010):

> hubo un tiempo en el que todos nosotros proclamamos la llegada de una "revolución electrónica", un tiempo en el que los artistas, científicos y pensadores en sintonía con su época creyeron que las computadoras y las redes telemáticas constituirían ciertamente el ambiente próximo de las nuevas formas culturales, o los motivos más apremiantes para un cambio radical de los propios conceptos de arte y de cultura. (p. 17)

Uno de los aspectos interesantes de ver a la computadora como un nuevo medio artístico es que esto responde a lo que Bolter y Grusin (2000) plantean en torno a lo que es nuevo en los nuevos medios, esto es la forma como ellos remodelan los medios más viejos y las maneras en que los medios más viejos se remodelan a ellos mismos para contestar los desafíos de los nuevos medios.

Un ejemplo de cómo el uso, desarrollo y experimentación con tecnologías libres empodera a los diseñadores y creadores pueden ser los proyectos desarrollados al interior del laboratorio Hipermedia de Tecnologías para la Comunicación de la Fundación Universitaria Los Libertadores sede Bogotá. Este es un lugar creado para jugar, experimentar y aprender con tecnologías (preferiblemente abiertas). Es un lugar para cometer errores, pero especialmente para aprender de los errores. En donde el conocimiento y los desarrollos que se hacen permiten potenciar e impulsar procesos en los contextos de la comunicación, de la creación y del entretenimiento. Es una iniciativa que se preocupa por explorar las posibilidades que pueden ofrecer diferentes tipos de tecnologías para la comunicación; de manera que diseñadores, comunicadores, publicistas, artistas o cualquier otro profesional interesado en procesos de comunicación e interacción pueda desarrollar estrategias comunicativas alternativas haciendo uso de tecnologías libres. Allí se han desarrollado proyectos en el contexto de videojuegos, de la robótica, de la visión por computadora, de las interfaces tangibles y el Internet de las cosas (Laboratorio Hipermedia, 2017).

Los artesanos del siglo XXI

Se puede considerar que los *hackers* y *makers* (hacedores), son los artesanos de finales del siglo XX y principios del XXI. Los *makers* son un ejemplo perfecto de *Design Thinking*. En Internet se encuentran infinidad de proyectos en los que los *makers* convierten bolsas plásticas llenas de agua con anilina y una webcam en superficies táctiles (Ellingsen, 2007) o proyectos en los que con botellas plásticas de agua construyen lentes para las gafas de realidad virtual *Google Cardboard* (Manual do Mundo, 2015).

Los *hackers* adaptan las tecnologías disponibles para que hagan otras cosas diferentes para las cuales fueron diseñadas inicialmente y liberan sus desarrollos como software libre. Un ejemplo de ello sería lo que hizo Johnny Lee (2008) con el *Wiimote* de Nintendo o lo que ha hecho Daniel Shiffman (2014) con el sensor *Kinect*.

Con los ejemplos anteriores se evidencia que los *hackers* y *makers* son artesanos por naturaleza, ya que "en la mente del artesano, la solución y el descubrimiento de problemas están íntimamente relacionados" (Sennett, 2009, p. 13-14).

Así pues, con la comprensión técnica que "se desarrolla a través del poder de la imaginación" (Sennett, 2009, p. 13), surge un arte que exige un gran "dominio técnico, ya no del lado del manejo de pinceles y martillos, sino, por ejemplo, en el novedoso campo del diseño y de la programación de software y de hardware" (Grisales, 2010, p. 204). Y es en este contexto, que el software libre se configura fuente de oportunidades creativas y como una alternativa de desarrollo e innovación a favor de la libertad de expresión.

Es entonces cuando el papel del creador "se reestructura con un mundo cambiante que exige otras competencias" (Fernández, 2013, p. 131), particularmente porque el "software expresa ideas" (Lanier, 2010, p. 6).

Así, el "gran desafío que la sociedad moderna debe afrontar: el de pensar como artesanos que hacen un buen uso de la tecnología" (Sennett, 2009, p. 33); ya que, como agrega Grisales (2010), "el técnico es pues, un creador" (p. 208).

Manifiesto de software libre para humanos

Esta arqueología muestra que, desde los primeros días de la computación, acceder al código fuente y poder modificarlo empoderaba a estos *hackers* responsables de darle forma a las tecnologías que usamos el día de hoy. No obstante, en la sociedad actual, nos encontramos frente a una gran amenaza y es que la propiedad intelectual se ha convertido en el petróleo del siglo 21 (Getty, 2000); tal como se establece en el *Remixer's Manifesto* (imagen 2), la propiedad intelectual buscará cada día tener mayor control y dominio sobre los aspectos básicos de la producción cultural y por ello para construir sociedades libres es necesario limitar el control del pasado (Gaylor, 2008).

Así pues, se hace imprescindible promover el uso y apropiación de las tecnologías libres como recursos alternativos y sustentables para asegurar que las generaciones del futuro tengan mejores oportunidades para acceder al conocimiento.

En este sentido sería válido proponer un nuevo manifiesto software libre (especialmente para humanos) dado que, como manifiesta Lanier (2010), esta ideología promueve una libertad radical en la superficie de la Web; aunque esa libertad, irónicamente, sea más para las máquinas que para las personas.

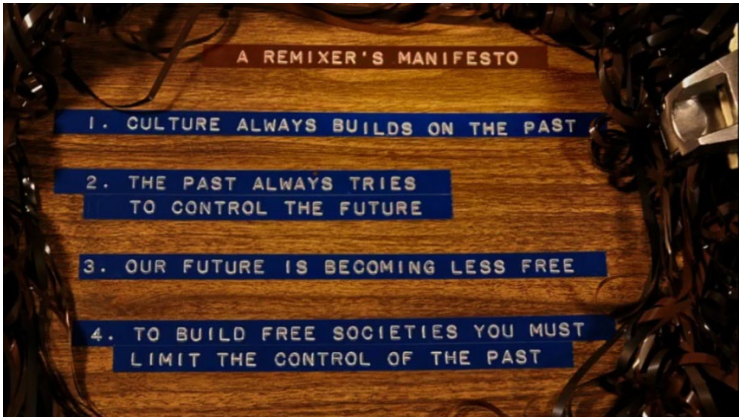


Imagen 2. RiP!: A Remixer's Manifesto.
Fuente: Gaylor (2008, 06:59).

Así, un manifiesto de software libre para humanos debe preocuparse porque cualquier usuario deba poder modificar el software que usa para adaptarlo a sus necesidades particulares. Se debe buscar, hacer cada vez más fácil la modificación del código (particularmente para los no expertos). Buscando ofrecer diferentes modos de visualización del código (ya sea texto, flujos, árboles o cualquier otro que surja). Y no solo se debe facilitar la escritura de código, pues sobre todo se debe tratar de facilitar la lectura de este. Se debe privilegiar la manipulación directa, de manera que los cambios hechos al programa se reflejen de forma inmediata. Que no se requiera recompilar los programas para ver si funcionan los cambios efectuados. Se debe garantizar que los no programadores puedan modificar y adaptar el software que usan. Se debe buscar que no se tenga que cambiar de hardware constantemente para acceder a las actualizaciones de software. Y, por último, se debe garantizar el derecho a modificar el hardware de manera que se le pueda dar usos insospechados a las tecnologías disponibles.

Algunas personas manifiestan su desacuerdo con que los artistas hagan cambios al software que usan; argumentan que esto podría repercutir en muchas versiones llenas de fallos y que esto es solo tarea de los ingenieros de sistemas o informáticos. No obstante este manifiesto de software libre para humanos no tiene que ver con la calidad del software, sino con poder ofrecer la posibilidad de ejercer la desobediencia tecnológica y la mal utilización de las tecnologías desde la perspectiva de Kurenniemi (como se citó en Taanila, 2003). De manera que al cambiar el software se puedan encontrar formas alternativas para expresar lo que sentimos, pensamos, queremos o perseguimos.

De algún modo esta postura es similar a la que proponía, en 1976, Nicholas Negroponte en: *Soft Architecture Machines*, sobre una "arquitectura sin arquitectos"; libro que ayudó a fundamentar la creación de lo que hoy se conoce como el Media Lab del MIT (Negroponte, 1976).

Conclusiones

Hoy en día, los creadores se ven cada vez más limitados por lo que indican las tecnologías de lo que se puede hacer o no. En el pasado, era el creador quien se veía obligado a desarrollar una nueva tecnología; si es que lo que quería hacer no se podía construir con las tecnologías disponibles a la fecha. Nos encontramos en un momento en el que no podemos permitir que los creadores nos convirtamos en simples creadores de contenidos con las tecnologías disponibles. Por ello, es muy importante que el desarrollo tecnológico esté al servicio de la creación y no al contrario.

Una de las principales preocupaciones de este artículo es la de advertir los riesgos de quedar atrapados en las jaulas que imponen las patentes y la innovación cerrada a los procesos de creación. Es importante encontrar estrategias que permitan empoderar a los artistas, diseñadores y creadores para

que puedan tener el control sobre las tecnologías y no sean las tecnologías las que les indique que pueden o no hacer. Y es allí, donde la innovación abierta y las tecnologías libres adquieren gran importancia.

Para lograr que los humanos puedan acceder de forma libre a las tecnologías de la electrónica y la computación, la fórmula debe ser una combinación de software libre, hardware abierto y estándares abiertos. La compaginación de estos tres elementos se podría definir como la fórmula ideal para construir un ecosistema favorable para alcanzar la autonomía, independencia y neutralidad tecnológica.

Una de las principales razones por las cuales muchos creadores y diseñadores no ven en el software libre una ruta alternativa para el ejercicio profesional, es debido a que muchas escuelas siguen solo adoptando tecnologías privativas con el fin de responder a las necesidades del mercado. Y, como lo manifiesta Victor, las tecnologías cambian muy rápidamente pero la mente de las personas cambia lentamente; así, mientras es más fácil adoptar nuevas tecnologías, en cambio, es mucho más difícil adoptar nuevas formas de pensar. Es en este sentido que es importante que los creadores y diseñadores dejen de pensar que el software libre es una opción no profesional, útil solo para los principiantes o aficionados; ya que el proceso creativo de un artista ya no está restringido por lo que dictan las compañías de software, sino por su propia habilidad y talento (Mansoux and Valk, 2008).

En definitiva, las tecnologías libres proporcionan una nueva e interesante ruta de acceso a una práctica alternativa del diseño debido a que le da al diseñador una gran cantidad de retos interesantes y de oportunidades creativas.

Referencias

Anderson, C. (2009). *Free: The future of a radical price*. New York, USA: Hyperion.

Barragán, H. (2003). *Wirign*. Recuperado de http://wiring.org.co/.

Bolter, J.D. and Grusin, R. (2000). *Remediation: Understanding new media*. Cambridge, USA: MIT Press.

Brand, S. (1988). *The Media Lab: Inventing the future at MIT*. New York, USA: Penguin Books.

Ellingsen, E. (2007). *$ 2 multitouch*. Recuperado de https://www.youtube.com/watch?v=yzNh31q61gc.

Elliott, J. (2008). *The Last HOPE: Dirty New Media - Art, Activism, and Computer Counter-Cultures*. Recuperado de https://www.youtube.com/watch?v=iFIwzB1WKz8.

Fernández, C. (2013). El diseñador del futuro. *KEPES*, *10* (9), 129-148.

Fogel, K. (2005). *Producing open source software: How to run a successful free software project*. Sebastopol, USA: O'Reilly.

Gates, B. (1976). *Una carta abierta a los usuarios por hobby*. Recuperado de http://alfonsogu.com/2007/04/27/carta-de-bill-gates-en-favor-del-software-propietario/.

Gaylor, B. (2008). *RiP!: A Remixer's Manifesto*. Recuperado de https://www.youtube.com/watch?v=Q-I5m3Sl_Gk.

Getty, M. (2000). *Blood and oil. The Economist*. Recuperado de

Gigliotti, D. (2003). *Radical Software*. Recuperado de http://www.radicalsoftware.org/e/history.html.

González, J., Seoane, J. y Robles, G. (2008). *Introducción al software libre*. Barcelona, España: Universitat Oberta de Catalunya.*KEPES*, *7* (6), 195-210.

Hancock, T. (2008). *License Classification Icons*. Recuperado de http://freedomdefined.org/Logos_and_buttons.

Higgins, D. (1970). *Computers for the arts*. New York, USA: Abyss publications.

Huhtamo, E. and Parikka, J. (2011). *Media Archaeology: Approaches, Applications, and Implications*. Chicago, USA: University of California Press.*You are not a gadget: A manifesto*. New York, USA: Vintage Books.

Lee, J. (2008). *Johnny Lee: Wii Remote hacks*. Recuperado de https://www.youtube.com/watch?v=QgKCrGvShZs.

Lessig, L. (2004). *Free culture: How big media uses technology and the law to lock down culture and control creativity*. New York, USA: Penguin Press.

Levy, S. (2010). *Hackers: Heroes of the Computer Revolution*. Sebastopol, USA: O'Reilly. Kac: cuerpos y mentes en expansión. En A. Burbano y F. Ali-Brouchoud (Ed.), *Eduardo Kac: el creador de seres imposibles* (pp.17-22). Manizales, Colombia: Universidad de Caldas.

Manrique, M.J. y Rueda, C.A. (2002). *Loro. Un sistema de programación*. Recuperado de https://sourceforge.net/projects/loro/.

Mansoux, A. and Valk, M. de (Ed.). (2008). *Floss + art*. Poitiers, France: GOTO 10.

Manual do Mundo. (2015). *Como fazer uma lente de aumento em casa*. Recuperado de https://www.youtube.com/watch?v=iGgO82eBsAI.

Morton, P. and Sandin, D.J. (1976). *General Motors (Part 2) - Phil Morton (1976)*. Recuperado de https://vimeo.com/59157626.

Negroponte, N. (1976). *Soft architecture machines*. Cambridge, USA: MIT Press.

Nelson, T. (2008). *Geeks Bearing Gifts*. Sausalito, USA: Mindful Press.

Nelson, T. (2012). *Computers for Cynics 0 - The Myth of Technology*. Recuperado de https://www.youtube.com/watch?v=ugcUNI9Wl2c.

Sennett, R. (2009). *El artesano*. Barcelona, España: Anagrama.

Shiffman, D. (2014). *OpenKinect-for-Processing*. Recuperado de https://github.com/shiffman/OpenKinect-for-Processing.

Stallman, R. (1983). *Initial Announcement - GNU Project*. Recuperado de http://www.gnu.org/gnu/initial-announcement.en.html.

Stallman, R. (1985). *The GNU Manifesto. Dr. Dobb's Journal, 10*. Recuperado de http://www.drdobbs.com/open-source/the-gnu-manifesto/222200498.

Stallman, R. (1986). *What is the Free Software*. Recuperado de https://www.gnu.org/bulletins/bull1.txt.

Stephenson, N. (2003). *En el principio... fue la línea de comandos*. Madrid, España: Traficantes de Sueños.

Taanila, M. (2003). *The Dawn of dimi*. Documental.

Victor, B. (2012). *Bret Victor - Inventing on a Principle*. Recuperado de https://www.youtube.com/watch?v=a-OyoVcbwWE&feature=youtu.be.

Victor, B. (2013). *Bret Victor The Future of Programming*. Recuperado de https://www.youtube.com/watch?v=8pTEmbeENF4.

Victor, B. (2013). *Bret Victor - Stop Drawing Dead Fish*. Recuperado a partir de https://www.youtube.com/watch?v=ZfytHvgHybA.

Zielinski, S. (1996). *Media Archaeology*. Recuperado de http://www.ctheory.net/articles.aspx?id=42.

Como citar este artículo: Cuartas Correa, J.D. (2018). Reflexiones en torno a la arqueología de medios en el software libre. *Revista KEPES*, *15* (17), 69-92. DOI: 10.17151/kepes.2018.15.17.4