Question
Revista Especializada en Periodismo y Comunicación
ISSN 1669 - 6581

Av. 44 Nº 676, 1º piso Of. 16
CP 1900 – La Plata – Argentina
Tel: +54 (221) 422-4090 / 423-6778
www.perio.unlp.edu.ar/question

Vol. 1, N.° 44 (octubre-diciembre de 2014)

DESARROLLO COLABORATIVO DE SOFTWARE LIBRE EN LA ARGENTINA: LA EXPERIENCIA DE UN *HACKATÓN* CORDOBÉS

Agustín Zanotti
Universidad Nacional de Villa María (Argentina)

Resumen
El artículo analiza una experiencia de creación colaborativa de software realizada en la Argentina. Se trató de un *hackatón* o maratón de programación, en el cual se trabajó durante cuatro semanas en el desarrollo de un controlador de red inalámbrica para el *kernel Linux*. La experiencia que presentamos permite descubrir varios elementos vinculados al modelo libre de desarrollo de software, el cual postula la *libertad* de acceso, uso y apropiación del código que conforma sus aplicativos informáticos.
A lo largo del trabajo analizamos el desarrollo de la experiencia y profundizamos sobre algunas de sus dimensiones: el trabajo de programación, las herramientas de construcción colaborativa, los valores y representaciones asociados al modelo libre y algunos de sus proyectos, las formas de sociabilidad y prácticas habituales de estos entusiastas informáticos, entre otras. Intentamos con ello rescatar la riqueza y complejidad de este tipo de episodios.

Palabras clave: hackatón, software libre, kernel, trabajo colaborativo.

Introducción

El presente artículo analiza una experiencia de creación colaborativa de software realizada en la Argentina. Se trató de un *hackatón* o maratón de programación, en el cual se trabajó durante cuatro semanas en el desarrollo de un controlador de red inalámbrica para el *kernel Linux* (1). La experiencia que presentamos permite descubrir varios elementos vinculados al modelo libre de desarrollo de software, el cual postula la *libertad* de acceso, uso y apropiación del código (2) que conforma sus aplicativos informáticos. Esto posibilita modificar el software, mejorarlo o adaptarlo para nuevos usos y compartir estas mejoras con la comunidad (Proyecto GNU, 2011).

El software libre plantea principios alternativos para producir y relacionarse con las tecnologías informacionales. Algunos de sus precursores lo definen como una forma "ética" de entender el software (Stallman, 2004). Este modelo genera una mayor autonomía, tanto para sus desarrolladores como sus usuarios, lo cual redunda en formas de organización de la producción más desconcentradas (Zanotti, 2011a). Los entornos virtuales posibilitan construir relaciones, trabajar y compartir actividades cotidianas, y se combinan a menudo con encuentros de copresencia que combinan componentes rituales, identitarios y celebratorios.

Las comunidades y organizaciones vinculadas al software libre presentan además varios atributos que nos permiten analizarlos en términos de lo que Melucci (1996) define como un *movimiento social contemporáneo*. Desde esta perspectiva identificamos la creación de lazos de *solidaridad,* a través de redes

Av. 44 Nº 676, 1º piso Of. 16
CP 1900 – La Plata – Argentina
Tel: +54 (221) 422-4090 / 423-6778
www.perio.unlp.edu.ar/question

y formas de organización diversas que promueven el trabajo y la creación colaborativa. Las demandas y disputas del movimiento libre manifiestan asimismo un *conflicto* en torno a las formas de apropiación del conocimiento y la información. Este se relaciona con una serie de transformaciones globales recientes vinculadas a la privatización de la informática y su configuración como industria global, la conformación de modelos comerciales concentrados y excluyentes que restringen ciertos usos posibles y el surgimiento de las redes virtuales como un nuevo espacio público (Zanotti, 2011b).

A lo largo del trabajo analizamos el desarrollo de la experiencia y profundizamos sobre algunas de sus dimensiones: el trabajo de programación, las herramientas de construcción colaborativa, los valores y representaciones asociados al modelo libre y algunos de sus proyectos, las formas de sociabilidad y prácticas habituales de estos programadores, entre otras. Intentamos con ello rescatar la riqueza y complejidad de este tipo de episodios.

El estudio se basa en el seguimiento del *hackatón* que tuvo lugar en la ciudad de Córdoba, Argentina, durante agosto de 2009. Para ello se realizaron observaciones participantes en diferentes encuentros que conformaron la actividad, complementadas con entrevistas a organizadores y el análisis de otros recursos disponibles on-line: entradas del blog VT6656 Linux Driver, la lista de distribución *kernel-hackathon,* así como un informe realizado al finalizar la actividad.

El *hackatón*

En el ámbito de la producción de software, y en especial en el software libre, se realizan a menudo actividades de desarrollo intensivo como *sprints* y *hackatones* (3). La palabra *hackatón* es una expresión compuesta por *hacker* y *maratón*. Se trata de encuentros concentrados de desarrollo de software en los que sus participantes se reúnen a programar de manera colaborativa, unidos por el interés hacia un determinado desarrollo y las posibilidades de aprender de *hackers* de mayor experiencia. El hecho de programar en situaciones de copresencia es sumamente valorado entre sus participantes. En nuestro medio local se han realizado varios encuentros de este tipo, siendo los *PyCamps*, campamentos de programación de lenguaje *Python*, los más difundidos.

En el caso del *VT6656 Linux Driver,* la realización del maratón surgió a partir de la posibilidad de contar con la presencia de un *hacker* de gran experiencia en el *kernel*, reconocido dentro de los veinte mayores desarrolladores mundiales (Linux Foundation, 2013). Se trató de la primera experiencia de este tipo en el medio local, "una oportunidad que no podía dejarse pasar" para el grupo de entusiastas que formaron parte de ella.

En su organización participaron miembros de la comunidad local GRULIC (Grupo de Usuarios de software libre de Córdoba) y de la Fundación Vía Libre. Ambos tomaron a su cargo las tareas de logística necesarias: conseguir el espacio, montar los servicios y la infraestructura informática requerida y convocar a los interesados. El encuentro se llevó a cabo en un *lab* de la Facultad de Matemática, Astronomía y Física de la

Av. 44 Nº 676, 1º piso Of. 16
CP 1900 – La Plata – Argentina
Tel: +54 (221) 422-4090 / 423-6778
www.perio.unlp.edu.ar/question

Universidad Nacional de Córdoba, con el apoyo de docentes que colaboraron en el seguimiento de las actividades.

En cuanto a los participantes, estos eran en general estudiantes avanzados o programadores con alguna experiencia en desarrollo de software. Todos se consideraban en mayor o menor medida usuarios de GNU/Linux. Solo una de las catorce personas era mujer y las edades del grupo variaban entre los 20 y los 45 años. Varias de las personas se conocían entre sí a partir de su pertenencia a la Universidad o su participación en comunidades libres locales.

Durante el desarrollo de la experiencia se realizaron, además, tareas de difusión en medios televisivos así como partes de prensa en periódicos locales. Junto a ello se montó el blog *VT6656 Linux Driver,* en el que se fueron socializando los avances y algunos aspectos técnicos. La lista de distribución *kernel-hackathon* se puso asimismo a disposición de los participantes para el intercambio y la comunicación interna. El proyecto incluyó una instancia final de evaluación por parte de los organizadores, cuyos resultados fueron incorporados en un informe. Luego del maratón, los participantes realizaron una presentación abierta al público en la que efectuaron un balance de la experiencia.

El desafío del *kernel*

El *kernel* o núcleo de los sistemas basados en GNU/Linux es un desarrollo que lleva más de veinte años de construcción acumulativa. Fue iniciado en 1991 por Linus Torvalds, mentor del emprendimiento hasta la actualidad y de quien tomaría su nombre. En nuestros días se considera uno de los mayores proyectos de programación existentes, así como una de las mayores experiencias de construcción colaborativa de software. Según el último informe disponible, el *kernel* (en su versión 3.10) se compone de 43.029 archivos y cuenta con casi 17 millones de líneas de código (Linux Foundation, 2013).

Su desarrollo continúa a un ritmo vertiginoso: entre 8 y 12 semanas es el tiempo promedio hasta el lanzamiento de su siguiente versión estable, que regularmente incorpora mejoras, nuevas funcionalidades y soporte de *hardware.* El *kernel* toma por base los principios del movimiento de software libre: su código fuente se publica junto a las versiones ejecutables, de manera de estar permanentemente disponible. El trabajo sobre el *núcleo* se nutre del trabajo de miles de colaboradores, que incluye tanto voluntarios como desarrolladores pagos, y recibe aportes de las mayores empresas del ámbito tecnológico (Linux Foundation, 2013).

Tuomi (2006) ha analizado la dinámica de este sistema como un proceso de *coevolución técnica y social*. Su crecimiento y reescritura gradual se acompaña de un número cada vez mayor de programadores incorporados al proyecto, que implica la coordinación de un gran número de personas con motivaciones diferentes. Las posibilidades de vinculación en red y las distintas herramientas de desarrollo disponibles, así como una suerte de contrato social entre sus colaboradores, sirven para la resolución de conflictos, la regulación y gestión del proyecto que busca mantener un equilibrio entre innovación, estabilidad y el control

Question
Revista Especializada en Periodismo y Comunicación
ISSN 1669 - 6581

Av. 44 Nº 676, 1º piso Of. 16
CP 1900 – La Plata – Argentina
Tel: +54 (221) 422-4090 / 423-6778
www.perio.unlp.edu.ar/question

de su complejidad interna.

La arquitectura del núcleo presenta una división modular con interfaces que permiten su interacción y posibilita la articulación de una gran cantidad de elementos. Sus desarrolladores pueden así especializarse en ciertas partes del código, sin tomar en consideración la totalidad de su estructura. Linux es, de este modo, una multiplicidad de desarrollos al interior de un proyecto mayor (Tuomi, 2006).

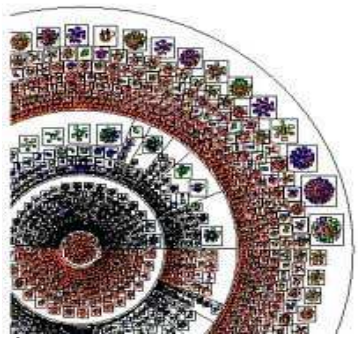

Figura 1:
Arquitectura del *kernel Linux*

El *kernel* recibe aportes provenientes de distintas regiones del mundo, aunque con diferencias notables en favor de ciertos países con mayor desarrollo tecnológico. De acuerdo con los participantes del *hackatón*, la Argentina cuenta con algunos desarrolladores:

> Por cierto, gente en la Argentina que programa en el kernel ha habido desde hace ya bastante tiempo. [...] hasta yo mismo he contribuido mi propio parche (4). Trivial, por supuesto, sin el menor asomo de creatividad, si vamos al caso ni siquiera califica de "programación", y ni me dio la cabeza para formatearlo bien, pero la sensación de haber contribuido un infinitésimo, aunque sea, a que el *kernel* ande mejor en un montón de máquinas, es difícil de describir más allá de decir que está muy bueno (VT6656 Linux Driver blog, 14/08/09).

La intervención sobre el *kernel*, aunque sea sobre elementos pequeños o triviales, se convierte en un gran desafío para muchos programadores. Existe además una serie de "leyendas" asociadas al sistema y la dificultad para iniciarse en las destrezas necesarias para comprenderlo, que a menudo tienden a disuadir o desanimar a aquellos principiantes que pretenden acercarse como colaboradores. Sus desarrolladores parecen constituirse en un grupo con talento especial y un fuerte compromiso hacia el proyecto:

> ... la leyenda dice que desarrollar para el *kernel* es reeeee difíiiiiicil y mucha gente se asusta antes de empezar. En muchos lados, este umbral psicológico es relativamente sencillo de superar, simplemente porque hay varios *hackers* de *kernel* cerca, con los que uno puede comer una pizza y darse cuenta de que son simples mortales como cualquier otro (aunque probablemente tomen más cerveza que el promedio) y que el *kernel* es un programa C como cualquier otro: grandote, y un poco más incómodo de *debuggear*, pero un programa más. Así, donde ya hay *hackers* del *kernel*, aparecen más *hackers* de *kernel* (VT6656 Linux Driver blog, 14/08/09).

Entre *hackers* y *geeks*

En los días previos al comienzo del *hackatón*, los organizadores pusieron a consideración del grupo mi participación en la experiencia a través de un mensaje encabezado "Sociologist watching geeks". Los

Av. 44 N° 676, 1° piso Of. 16
CP 1900 – La Plata – Argentina
Tel: +54 (221) 422-4090 / 423-6778
www.perio.unlp.edu.ar/question

participantes a menudo se autodenominaban *geeks*, una expresión común en la jerga de los informáticos referida a sujetos atraídos y hasta fascinados por las tecnologías. El gusto por la ciencia ficción, los juegos de rol y cómics, un aspecto informal y sedentario, una destreza para las ciencias así como una cierta dificultad en sus relaciones sociales son algunos de los aspectos que completan el estereotipo *geek*. Incluso el propio creador del sistema Linux se refiere a sí mismo de este modo:

> Yo era un *nerd*. *Geek*. Desde el comienzo. No pegaba con cinta mis gafas, pero bien podría haberlo hecho, porque tenía todas las otras características. Bueno para las matemáticas, bueno en física y sin gracias sociales de ningún tipo. Y esto fue antes de que ser un *nerd* fuese considerado algo bueno (Torvalds y Diamond, 2001: 4 [traducción propia]).

Algunas definiciones coinciden en que el término *geek* estaba hasta hace poco tiempo asociado a connotaciones negativas o peyorativas (Jargon File, 2011). Más recientemente, de la mano de la creciente centralidad de las tecnologías informacionales en los procesos económicos y sociales, se ha producido una revalorización de este tipo de perfiles:

> Cuando la computación evolucionó hacia su masificación, los *geeks* comenzaron a adoptar un decisivo rol tecno-social [...]. Hoy, son los *geeks* quienes a su modo rediseñan la economía convirtiéndola en *geekonomía*. A su vez son ellos quienes configuran los formatos de nuestras relaciones personales. Desde la irrupción de Internet y la computación personal, los *geeks* son los nuevos escribas del mundo, capaces de crear los instrumentos que utilizan, o apropiarse de manera especial de los ya creados (Pardo Kuklinsky, 2010).

En cierta proximidad con lo anterior, el término *hacker* se refiere a una forma particular de concebir el trabajo de los informáticos. Se asocia a la pasión por la programación, el reconocimiento de la creatividad y el mérito y, fundamentalmente, una *ética* de compromiso por la libertad de la información (Himanen, 2002). Algunas definiciones más ambiciosas hacen alusión a una *cultura hacker* que incluye intereses, hábitos, maneras de vestir, un léxico que los caracteriza y hasta orientaciones políticas y religiosas (Jargon File, 2013).

Durante el transcurso del taller, el coordinador era definido como *hacker*, indicando de alguna manera un reconocimiento de su experiencia por parte del resto del equipo. Este demostraba una increíble facilidad para interpretar el código y encontrar errores: “Lo que hace este tipo es muy difícil, debe tener una gran experiencia reparando código”, destacaba uno de los participantes. Más allá de

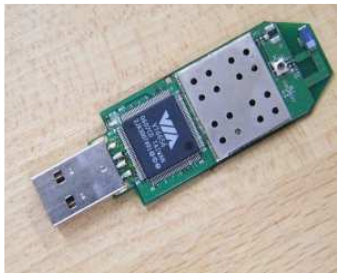

Figura 2:
VT6656 USB Wi-Fi Adapter

Question
Revista Especializada en Periodismo y Comunicación
ISSN 1669 - 6581

Av. 44 N° 676, 1° piso Of. 16
CP 1900 – La Plata – Argentina
Tel: +54 (221) 422-4090 / 423-6778
www.perio.unlp.edu.ar/question

ambas definiciones, el entusiasmo y la pasión por la experimentación con tecnologías quedaron evidenciados en los encuentros así como en ciertos registros del blog:

> Electropornografía:
> Viendo que el gabinete de plástico en el que contiene el dongle se mantiene cerrado con dos tornillos en miniatura, no pudimos resistirnos a "desvestirlo" para poder admirar la electrónica en toda su gloria. En la cara que queda para arriba cuando se enchufa el dongle en una máquina pueden verse el VT6656 y la antena. El aparato de radio está, pero no se ve porque está debajo del blindaje. Entre el blindaje y la antena hay un trimmer de función desconocida (VT6656 Linux Driver blog, 14/08/09).

La experiencia de desarrollo

Como anticipamos, el objetivo del *hackatón* fue la construcción de un controlador para un dispositivo de red inalámbrico portátil. Un controlador o *driver* es lo que permite que los equipos reconozcan sus diferentes piezas de hardware y puedan utilizarlas. Al finalizar el taller, este sería incorporado a la rama de desarrollo del *kernel*.

El soporte para dispositivos en GNU/Linux suele ser un tema de controversias (5). Esto ha conducido a que en varias oportunidades surjan intentos de construir comunitariamente ciertos controladores para dar soporte a los usuarios. Tales esfuerzos acarrean una elevada complejidad y en varias ocasiones suponen ejercicios de *ingeniería inversa,* un método de resolución supone profundizar el estudio del dispositivo hasta el punto en que puedan llegar a reconstruirse las instrucciones que determinan su funcionamiento.

En el caso del *hackatón*, los adaptadores en cuestión no contaban hasta el momento con soporte para sistemas GNU/Linux y la empresa fabricante no tenía planeado desarrollar los controladores. El equipo se contactó, sin embargo, con la firma para ponerla al tanto del proyecto y logró que donara los dispositivos requeridos y aportara algunas especificaciones necesarias.

Con los objetivos trazados, se estableció un cronograma de trabajo y se empezaron a poner en común algunos de los requerimientos básicos para realizar las tareas. En todas las sesiones se debería cumplir con un avance hacia la construcción del *driver*, pasando por una sucesión de etapas técnicas del desarrollo. Las tareas pendientes deberían continuarse en los días intermedios hasta el próximo encuentro.

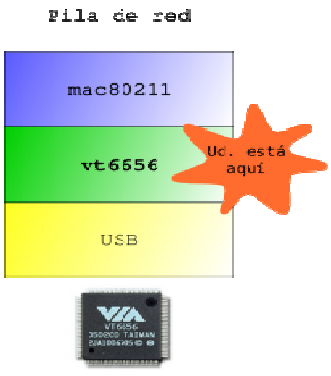

Figura 3:
Etapas sucesivas del desarrollo

El trabajo de programación consistía básicamente en escribir líneas de código de acuerdo con una serie de reglas correspondientes al lenguaje utilizado. Sobre un programa similar a un editor de textos, los elementos aparecían en diferentes colores para facilitar su

Av. 44 N° 676, 1° piso Of. 16
CP 1900 – La Plata – Argentina
Tel: +54 (221) 422-4090 / 423-6778
www.perio.unlp.edu.ar/question

identificación. Los programadores se manejaban rápidamente sobre el código, borrando, escribiendo y modificando diferentes parámetros. Se trataba de una tarea que requería atención a la coherencia entre las partes y su lógica interna.

Las soluciones a los diferentes problemas no eran únicas y a menudo algunas solían ser más "prolijas" y acertadas que otras. La escritura revestía así un cierto carácter *artesanal.* Se incluían además en el código mensajes que indicaban el uso de determinados fragmentos, de manera de simplificar su lectura para los demás programadores.

Una parte del trabajo consistió en ejercicios recurrentes de ensayo y error. Las nuevas instrucciones se iban probando sobre el dispositivo para verificar su funcionamiento y observar cómo reaccionaba ante cada modificación. En la pantalla podían verse diferentes resultados. A menudo aparecía algún tipo de error, que los programadores interpretaban de diferentes formas. La tarea parecía compleja y desafiante. En la medida en que las soluciones posibles eran múltiples, las versiones de código que manejaban los programadores necesitaban, cada cierto tiempo, ser unificadas. Las mejores soluciones eran implementadas y compartidas con el grupo, al tiempo que el coordinador animaba a los participantes a encontrar alternativas "más inteligentes".

Los participantes se mantuvieron hasta el final de la experiencia. La complejidad del proyecto terminó por requerir más tiempo que el pautado originalmente y los programadores dedicaron horas extras durante la noche o los fines de semana para intentar resolver algunas situaciones no previstas. En las últimas reuniones la dinámica de trabajo cambió, y los programadores decidieron dividirse las tareas remanentes y trabajar en grupos.

Algunas cuestiones relacionadas con la transmisión y la recepción de señales, necesarias para que el dispositivo finalmente lograra funcionar, resultaron más complejas de lo esperado y llevarían más trabajo del originalmente previsto. Era claro que no lograrían finalizarlas a tiempo. El equipo probó un par de soluciones y se discutieron posibles líneas a seguir. Los organizadores propusieron tomar una semana más y realizar, para concluir, una presentación abierta al público para dar a conocer los resultados del proceso.

Herramientas y dinámicas colaborativas

Para avanzar en el desarrollo, era necesario que todos los participantes conocieran ciertas herramientas y que estuvieran familiarizados con la modalidad de trabajo utilizada en este tipo de proyectos:

> Así, dedicamos todo el encuentro a asegurarnos de que todos tenemos: 1. La misma versión de *kernel* (2.6.30.4, compilado con el mismo archivo .config en todas las máquinas). 2. El sistema de control de revisiones *git*, que es el usado para desarrollo de Linux, y sus herramientas asociadas. 3. El programa sparse, que hace verificación semántica de programas C… esencialmente un programa cuya tarea es prestar atención a que no cometamos cierto tipo de errores (VT6656 Linux Driver blog, 11/08/09).

Av. 44 N° 676, 1° piso Of. 16
CP 1900 – La Plata – Argentina
Tel: +54 (221) 422-4090 / 423-6778
www.perio.unlp.edu.ar/question

El trabajo con *git,* un sistema de gestión y control de revisiones distribuidas, permitía a los programadores trabajar en equipo e integrar las modificaciones realizadas sobre el código. Todos debían utilizar una idéntica versión del *kernel* en sus computadoras. Todo el software requerido se encontraba disponible en diferentes repositorios libres.

Los encuentros funcionaban como una instancia central de producción e intercambio entre los programadores. Luego, una parte importante de la interacción circulaba a través de la lista *kernel-hackatón* creada para tal fin. Esta lista, albergada en los servidores de la comunidad local, servía para discutir los avances y poner en común las modificaciones realizadas al código. Su actividad era intensa y suponía de parte de los participantes el conocimiento de ciertos códigos de conducta que regulaban las interacciones.

Era necesario "aprender a comportarse como un buen desarrollador", lo cual implicaba el seguimiento de ciertas pautas de "buena educación". No ser descortés, no alterarse ante las críticas despiadadas de los demás y responder solo con argumentos técnicos eran algunas de las reglas enunciadas. Otra cuestión fundamental radicaba en el envío de *parches* o *patchs*:

> Mientras tanto, vamos aprendiendo otras reglas del arte de la comunicación en grandes proyectos. Por ejemplo: nunca "relatar" cómo cambiaría uno el código, lo mejor es sencillamente cambiarlo y enviar un parche para que otros puedan verlo. El parche es la unidad elemental de intercambio de código entre desarrolladores, y hay una serie de reglas para enviarlos (VT6656 Linux Driver blog, 14/08/09).

Los *parches* son archivos, en general pequeños, con modificaciones que se aplican al código. Indican las líneas que deben ser agregadas, modificadas o removidas con la finalidad de corregir errores, agregar funcionalidades o mejorar las ya existentes (Linux Foundation, 2012). La circulación de los parches en la lista de correo debía seguir las siguientes pautas:

> 1. los parches debían ser la solución completa a un problema;
> 2. no se debía enviar más de uno por correo;
> 3. el asunto del mensaje debía indicar que este contenía un "[PATCH]";
> 4. estos debían contener una línea "Signed-off-by:" indicando el nombre de su creador;
> 5. los parches debían respetar cuidadosamente las convenciones de codificación del *kernel*
> (VT6656 Linux Driver blog, 14/08/09).

Estos recursos y pautas posibilitaban coordinar las tareas de programación entre los miembros del equipo y continuar los trabajos por separado luego de los encuentros.

Av. 44 Nº 676, 1º piso Of. 16
CP 1900 – La Plata – Argentina
Tel: +54 (221) 422-4090 / 423-6778
www.perio.unlp.edu.ar/question

Vol. 1, N.° 44 (octubre-diciembre de 2014)

Diversión, libertad, comunidad

La *vocación*, *satisfacción* y *adrenalina* asociada a la escritura de código ha sido destacada en diferentes estudios acerca de los trabajadores informáticos (Montes Cató, 2010: 81). Estos elementos se observan en el caso del software libre, respecto del compromiso y la dedicación voluntaria hacia este tipo de tareas. El *hackatón* representó para los participantes una oportunidad de socializar con personas con un mismo interés, adquirir nuevas destrezas de la mano de *hackers* de gran experiencia y compartir saberes. La posibilidad de "programar en libertad", de forma voluntaria, colectiva y colaborativa, en circunstancias elegidas por ellos mismos y sin presiones ni condicionamientos externos eran un fuerte elemento motivador.

En algunos encuentros los programadores proponían apagar las luces del *lab* para mejorar su concentración, quedando iluminados únicamente por las pantallas de las *laptops*. Los que lograban resolver su cometido podían asimismo aprovechar para explorar diferentes aspectos del dispositivo y experimentar libremente con funcionalidades desconocidas. El coordinador tenía un trato informal y distendido con los participantes, mezclaba cuestiones técnicas con humoradas y anécdotas.

Más allá de todo esto, tal como señala el propio iniciador del *kernel*, una de las principales motivaciones de los *hackers* es la *diversión* (Torvalds y Diamond, 2001). Esta se relaciona con el hecho mismo de programar, con la posibilidad de experimentar con tecnologías novedosas, lidiar con problemas complejos y desarrollar la creatividad. Los participantes aludían a este componente de diversión en el caso del *hackatón*. Mientras cada uno se mostraba celoso de su propio trabajo, el intercambio de pedazos de código con soluciones y recomendaciones se fue haciendo más fluido a medida que avanzaban los encuentros. Se suscitó así una forma de competencia en clave de juego por intentar resolver exitosamente ciertas tareas complejas.

Pero no todo fueron tareas de programación. Las salidas "por unas cervezas" se convirtieron en una suerte de rutina posterior a las sesiones, a las que podían sumarse otras reuniones entremedio de los encuentros. Los avances sobre el proyecto constituían un motivo de celebración. Con el tiempo el trato se hacía más informal: "Te explico si me pagas una cerveza" decía un miembro del equipo, apoyando una tradición de alto consumo de esta bebida que me fuera referida en varias oportunidades. Se iba creando así un vínculo que trascendía las actividades formales. Los organizadores alentaban este tipo de prácticas y buscaban que los participantes compartieran tiempo con el coordinador para ayudarlo a conocer la cuidad y ponerlo al tanto de las costumbres locales.

Tal como lo ha destacado Coleman (2010) a propósito de otras experiencias *hackers*, el hecho de congregar a personas con intereses e identidades compartidas termina por redundar en un "encantamiento de la experiencia":

> Las conferencias de *hackers* son rituales de confirmación, liberación, celebración y en especial reencantamiento, donde los asuntos cotidianos de la vida, el trabajo y las interacciones sociales son ritualizados y por lo tanto experimentados en términos fundamentalmente diferentes. A través de una

Av. 44 Nº 676, 1º piso Of. 16
CP 1900 – La Plata – Argentina
Tel: +54 (221) 422-4090 / 423-6778
www.perio.unlp.edu.ar/question

> celebración condensatoria, los *hackers* imbuyen sus acciones de nuevos significados, revitalizados o éticamente recargados [...]. Se trata de profundos momentos de reencantamiento cultural, en donde los participantes construyen y comparten una experiencia engrandecida de sí mismos (Coleman, 2010: 53 [traducción propia]).

Como resultado obtuvimos una combinación intensa entre trabajo colaborativo, socialización de conocimientos, creación y celebración de comunidad, que caracterizó la experiencia hasta el final.

El balance de la experiencia: "no solo del código vive el *hacker*" (6)

El *hackatón* alcanzó a completarse y se avanzó significativamente en el desarrollo del controlador. La propuesta buscó asimismo generar un "efecto demostración" que diera cuenta de que era posible concretar proyectos de alto perfil sobre este tipo de tecnologías en el medio local. Se trató en este sentido de una experiencia particular de desarrollo de software que sentó un precedente en la región. Si bien los organizadores celebraron los objetivos alcanzados, lamentaron que no llegase a constituirse a largo plazo un equipo de trabajo que perdurara más allá del encuentro (FLOSSWorld, 2010).

La experiencia dejó a su vez al descubierto algunas dificultades. Entre ellas, destacó la dificultad de vinculación con los fabricantes de dispositivos. Los participantes mencionaron, además, la gran demanda de tiempo requerida por el proyecto, la falta de liderazgo del equipo luego de concluido el evento y el escaso interés por el producto final que se estaba generando (FLOSSWorld, 2010). Encontramos de este modo problemas similares a los de otros proyectos de programación de bajo nivel, los cuales revisten simultáneamente una elevada complejidad y una escasa notoriedad por parte de los usuarios. Por otra parte, el interés de los participantes se mostró más centrado en el proceso de creación, el encuentro y el aprendizaje que en el resultado de ese proceso.

A lo largo del trabajo buscamos reconstruir la experiencia del *hackatón* y resaltar algunas de sus dimensiones vinculadas a su organización, la movilización de recursos y las motivaciones que nucleaban a sus participantes. Como pudimos observar, más allá de las consideraciones técnicas, encontramos un conjunto de elementos significativos que nos permiten caracterizar este tipo de encuentros. La autopercepción de sus participantes como *hackers* o *geeks*, las representaciones que acompañan el desarrollo del *kernel* y a sus creadores, la pasión por las tecnologías y el desafío de afrontar tareas complejas, el reconocimiento del mérito acumulado por los participantes, los componentes lúdicos y solidarios que conllevan los episodios de trabajo colaborativo, las pautas de comportamiento y modos de hacer que orientan los intercambios entre programadores, la intensidad de los encuentros de copresencia, la proliferación de espacios de participación y creación colectiva, así como la multiplicidad de motivaciones que se vehiculizan y refuerzan en las comunidades libres son algunos de los elementos que componen el mundo cotidiano de estos entusiastas informáticos, que forman parte del modelo libre de producción de

Av. 44 Nº 676, 1º piso Of. 16
CP 1900 – La Plata – Argentina
Tel: +54 (221) 422-4090 / 423-6778
www.perio.unlp.edu.ar/question

software.

Mediante procesos de construcción orientados por valores, la libre disponibilidad del código fuente producido, la conformación de equipos de trabajo colaborativo en red y una serie de mecanismos que posibilitan multiplicar y gestionar una mayor diversidad de proyectos y motivaciones, el modelo libre está logrando redefinir el horizonte de desarrollo de la industria informática en su conjunto.

Notas

(1) Nos referimos al *kernel* o núcleo de los sistemas basados en Linux, el cual permite el funcionamiento de una amplia variedad de dispositivos electrónicos, desde teléfonos móviles, tabletas, PC, enrutadores, consolas de video-juegos, hasta computadoras de escritorio y servidores. Linux es el sistema más utilizado en servidores y en las supercomputadoras más poderosas en el mundo (Linux Foundation, 2012).

(2) Nos referimos al *código fuente*, el cual contiene las instrucciones escritas en lenguaje de programación y se considera como el "texto original" del programa, previo a ser compilado para poder ser utilizado de manera ejecutable. Este permite su inteligibilidad, su aprendizaje y su intervención para realizar modificaciones o elaborar futuras piezas de software.

(3) *Sprint* (en inglés) y *hackatón* son palabras derivadas de la jerga del deporte, en particular del atletismo. Mientras que el primero se refiere a competencias de distancias cortas, el segundo se corre sobre una mayor extensión.

(4) Como veremos, las contribuciones se organizan en forma de parches que pueden tener diferentes niveles de complejidad.

(5) Mientras en la actualidad Linux ha crecido en usuarios y en soporte, hasta hace no mucho tiempo los problemas con el *hardware* se contaban entre los principales obstáculos en la difusión hacia usuarios no expertos. Mientras que algunas empresas fabricantes ofrecen soporte de manera habitual, otras se resisten por diferentes motivos a brindar los controladores y especificaciones requeridos. En ciertos casos en los que existen controladores disponibles para los dispositivos, estos no satisfacen sin embargo a los usuarios o desarrolladores. Ello puede deberse a que sus funcionalidades son limitadas con relación a otros sistemas operativos o a que los controladores no cumplen con ciertas condiciones, siendo su código privativo o cerrado.

(6) Expresión tomada textual de la entrada del blog *VT6656 Linux Driver*, 14/08/09.

Bibliografía

Coleman, G. (2010), "The Hacker Conference: A Ritual Condensation and Celebration of a Life world", *Anthropological Quarterly*, Vol. 83, N.° 1, United States: George Washington University Institute for Ethnographic Research.

FLOSSWorld (2010), "Lessons learned from FLOSSWorld developer survey in Latin America", *D4.2 Validation Report FVL on Low participation in FLOSS development*, FLOSSWorld.

Himanen, P. *et ál*. (2002), *La ética del hacker y el espíritu de la era de la información*, España, Destino.

Linux Foundation (2012), *How Linux is Built* [en línea]. Disponible en: <www.youtube.com/watch?v=yVpbFMhOAwE> [Consulta: 11/07/13].

Linux Foundation (2013). *Linux kernel Development. How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It: September 2013*, Kroah-Hartman, G.; Corbet, J. y A. McPherson, US: Linux Foundation [en línea]. Disponible en: <http://go.linuxfoundation.org/who-writes-linux-2012>

Av. 44 Nº 676, 1º piso Of. 16
CP 1900 – La Plata – Argentina
Tel: +54 (221) 422-4090 / 423-6778
www.perio.unlp.edu.ar/question

[Consulta: 19/01/14].

Melucci, A. (1996), *Challenging codes. Collective action in the information age,* New York, Cambridge University Press.

Montes Cató J. S. (coord.) (2010), *El trabajo en el Capitalismo Informacional. Los trabajadores de la industria del software,* Buenos Aires, Poder y Trabajo Editores.

Pardo Kuklinsky, H. (2010), *Geekonomía. Un radar para producir en el postdigitalismo,* Barcelona, Publicacions i Edicions de la Universitat de Barcelona.

Stallman, R. (2004), *Software libre para una sociedad libre,* Madrid, Traficantes de Sueños.

Torvalds, L. and D. Diamond (2001), *Just for Fun: The Story of an Accidental Revolutionary,* US, HarperCollins Pub.

Tuomi, I. (2006), *Networks of Innovation. Change and Meaning in the Age of the Internet*, New York, Oxford University Press.

Zanotti, A. (2011a), "Reescribiendo tecnologías: Aproximaciones al movimiento software libre y su difusión en Argentina", *Intersticios. Revista sociológica de pensamiento crítico*, Vol. 5 (2) [en línea]. Disponible en: <www.intersticios.es/issue/view/803> [Consulta: 18/08/13].

Zanotti, A. (2011b), "Explorando el informacionalismo: nuevos escenarios de dominación, nuevos escenarios de disputa", *Astrolabio*, Vol. 7 (1) [en línea]. Disponible en: <revistas.unc.edu.ar/index.php/astrolabio/> [Consulta: 12/01/13].

Sitios web

The Jargon File: Geek. <http://www.catb.org/jargon/html/G/geek.html> [Consulta: 14/07/13].

The Jargon File: Hacker. <http://www.catb.org/jargon/html/G/hacker.html> [Consulta: 14/07/13].

Blog VT6656 Linux Driver. *Developing a Linux Driver for VIA's VT6656 USB Wi-Fi Adapter* <http://vt6656.wordpress.com> [Consulta: 14/07/13].

"kernel-hackathon" mailing list. <http://mx.grulic.org.ar/lurker/list/kernel-hackathon.en.html> [Consulta: 14/07/13].

PROYECTO GNU, 2011, *La Definición de Software Libre*. <www.gnu.org/philosophy/free-sw.es.html> [Consulta: 11/06/13].

Artículo recibido el 30/10/14 - Evaluado entre el 31/10/14 y 30/11/14 - Publicado el 21/12/14