XXVIII Congreso de la Asociación Latinoamericana de Sociología (ALAS).
Universidade Federal de Pernambuco, Recife, 2011.

# Hackeando el núcleo: una experiencia de desarrollo de software libre en Córdoba, Argentina.

Zanotti Agustín.

**XXVIII Congresso ALAS "Fronteiras Abertas da América Latina"**
**Recife – 06 a 10 setembro de 2011, UFPE, Recife-PE**
**GT01 - Ciência, Tecnologia e Inovação**

# Hackeando el núcleo: una experiencia de desarrollo de software libre en Córdoba, Argentina.

Agustín Zanotti
CIECS UNC-CONICET[1]
agustinzanotti@gmail.com

## Introducción

El presente trabajo analiza una experiencia de creación colaborativa de software libre realizada en Córdoba (Argentina) durante el año 2009. Se trató del *vt6656 kernel hackathon,* que tuvo por finalidad el desarrollo de un controlador para dar soporte en sistemas GNU/Linux[i] a un dispositivo de red inalámbrica.

La palabra *hackatón* (en inglés *hackathon*) es una expresión compuesta por "*hacker"* y "*maratón".* Se refiere así a una *maratón de programación o hacking,* eventos en los que, como veremos, un grupo de informáticos entusiastas se reúnen a *escribir código* de manera intensiva durante días e incluso semanas. Las personas trabajan de manera voluntaria y colaborativa, programando por diversión y en función de un objetivo común, al tiempo que se socializan saberes y experiencias.

Este tipo de manifestaciones, entre otras, caracterizan al modelo libre de producción de software, asociado a diferentes comunidades y organizaciones de software libre. Por oposición al modelo propietario[ii], este enfatiza el carácter de bien común del software, entendiéndolo como un conjunto de herramientas y conocimientos que deben administrarse con criterios no restrictivos y en función del bienestar de los usuarios y la sociedad en general. El modelo libre otorga la libertad para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software (www.gnu.org*, 2011).* Esto se hace posible gracias a que su *código fuente*[iii] se encuentra abierto (*open source*) y disponible en el dominio público.

La experiencia que presentamos permite descubrir la varios elementos vinculados a la producción de software y de software libre en particular: el trabajo de programación, las técnicas y herramientas de construcción colaborativa, los valores y representaciones asociados al software libre y las representaciones en torno a ciertos proyectos, las

1 Centro de Investigaciones y Estudios sobre Cultura y Sociedad. Universidad Nacional de Córdoba – Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina.

formas de sociabilidad y prácticas habituales entre los programadores, entre otras. A lo largo de la exposición analizaremos el desarrollo de la experiencia, profundizando sobre algunas de sus dimensiones. Con ello intentaremos rescatar la complejidad de este tipo de manifestaciones de creación y trabajo colaborativo en el software libre.

**"*Sociologist watching geeks*"**

Brindamos para comenzar algunas precisiones acerca del abordaje realizado. El mismo se basó en el seguimiento de la experiencia del hackatón realizado en Córdoba en Agosto de 2009. Para ello se utilizaron una diversidad de elementos disponibles, comenzando por instancias de observación participante en los diferentes encuentros que conformaron la actividad.

En cuanto a las instancias de observación, me enteré de la realización del maratón a partir de un grupo de distribución de la universidad nacional. La propuesta parecía bastante distendida: "*si vas a estar en Córdoba durante el mes de agosto y tenés ganas de pasartelá programando y tomar unas cervezas con Christoph, entonces sumate al taller*" (Registro de campo, primer encuentro). A partir de allí decidí contactar a uno de los organizadores para plantear la posibilidad de incorporarme. Me presenté como un sociólogo que estaba realizando investigaciones sobre temas relacionados al software y al software libre en particular. Luego de una puesta en común entre los organizadores me fue permitida la participación, la cual quedó sujeta a que no interfiriera en el desempeño de las actividades.

A partir de allí me sumé a los encuentros, con una muy buen disposición de parte de los organizadores. Al llegar el primer día fui presentado con Christoph, el *hacker* alemán que coordinaría la experiencia. Enseguida me preguntó si hablaba inglés, ya que los intercambios serían de allí en más en esa lengua, y si estaba cursando algún doctorado, a lo que asentí. Durante el primer encuentro, a poco de habernos presentado entre los participantes, este dejó traslucir un correo electrónico que se titulaba "*Sociologist watching geeks*". La expresión resume en cierta medida el vínculo creado a lo largo de la experiencia y resulta interesante en varios sentidos que desarrollaremos a lo largo de la exposición.

Mi participación consistió concretamente en formar parte de la mitad de los encuentros de tipo taller en que los programadores se reunían a escribir código fuente e intercambiar sus avances al respecto. Al no poseer conocimientos técnicos, resultaba difícil seguir este aspecto del proceso. Intenté valerme de las explicaciones que me

brindaban los participantes, así como de intercambios posteriores. Mi intervención en el espacio fue tomada con cierta naturalidad entre los participantes luego de un primer momento de inquietud. La misma perecía al mismo tiempo resaltar la trascendencia de lo que estaba por acontecer.

Las observaciones registradas se complementaron con otros recursos que posibilitaron comprender la complejidad de los intercambios y los procesos que se estaban realizando en el hackatón, así como las implicancias para sus participantes. Para ello se realizó una entrevista al coordinador de la maratón y se incorporó el *blog* (VT6656 Linux Driver) y la *lista de distribución* ("kernel-hackathon" mailing list), ambos creados a los fines de la difusión de la experiencia y la comunicación entre los participantes. Se analizó asimismo el informe final del taller realizado por los organizadores, el cual da cuenta de la propia perspectiva de los actores acerca de algunos aspectos determinantes y el balance de los resultados obtenidos. Todos estos materiales se encuentran disponibles en-línea en la actualidad.

**El Hackatón: "una oportunidad que no podemos dejar pasar[iv]"**

Tal como lo mencionaba uno de los participantes, la experiencia constituía una oportunidad que no podía dejarse pasar. Esto es así por varias razones. El hackatón se llevó a cabo en un *lab* de la Facultad de Matemática, Astronomía y Física (FAMAF) de la Universidad Nacional de Córdoba. En su organización participaron diferentes colectivos vinculados al software libre de trascendencia en el medio local: GRULIC (Grupo de Usuarios de software libre de Córdoba) y Fundación Vía Libre. Entre ambas tomaron a su cargo las diferentes tareas de logística que implicó la realización de la experiencia: contactarse con quien sería el coordinador, conseguir el espacio donde se realizaría la actividad, montar servicios informáticos para conectar a los participantes, difundir el evento y convocar a los interesados, recibir las solicitudes, entre otras tareas. También colaboraron dos docentes de FAMAF en el seguimiento de las actividades.

Por una cuestión de disponibilidad de espacio y de requisitos técnicos (los participantes tenían que disponer de su propia computadora portátil, se necesitaba un dispositivo por persona para trabajar), de las 25 personas que se mostraron interesadas en la propuesta finalmente pudieron incorporarse sólo 14. Los organizadores seleccionaron así a los participantes en función de los currículos previamente solicitados.

El proyecto contó además con el apoyo de la empresa fabricante de los dispositivos,

*Via Technologies Inc*., la cual donó a la Universidad los 12 adaptadores de red modelo *vt6656* sobre los que se trabajaría, y aportó algunas especificaciones necesarias para empezar el desarrollo del controlador.

Junto con el taller se pusieron en marcha diferentes actividades para dar difusión a la experiencia. Se montó un blog en el que se irían narrando los encuentros, se mostrarían los avances y algunos aspectos técnicos relacionados con el dispositivo. Los participantes podían postear allí sus comentarios e incluir nuevos temas de discusión. Se intentaron además otras vías de difusión del taller tales como entrevistas en medios televisivos y partes de prensa a periódicos locales. El proyecto incluyó asimismo una instancia de evaluación por parte de los organizadores, que constó de un cuestionario posterior al taller en el que se solicitaba a los participantes que realizaran un balance de la experiencia. Sus resultados fueron incorporados en un Informe presentado luego del final de la maratón y presentado en el proyecto FLOSSWorld (2010).

La realización del hackatón surgió a partir de la posibilidad de contar con la presencia de Christoph Hellwig, un programador alemán con gran experiencia en el kernel GNU/Linux, situado entre los 20 mayores desarrolladores a nivel mundial (Linux Foundation, 2010). Christoph ha trabajado en diferentes proyectos de software libre desde hace más de 10 años y se especializó en sistemas de archivos y almacenamiento, entre otros elementos. En los últimos tiempos se desempeña como *auto-empleado* haciendo consultorías, *contracting* y capacitaciones relacionadas con el núcleo y otros temas (Entrevista a Hellwig, 2009).

El *hacker* permanecería más de un mes en Argentina, por lo que su estadía brindaba un marco especial de posibilidad. Por medio del contacto con uno de los organizadores, surgió rápidamente la idea de realizar el taller. Su experiencia era reconocida por los demás participantes del hackatón. Este tenía 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*", decía uno de ellos. Más allá de eso, tenía un trato bastante informal y relajado. Siempre a horario en los encuentros, mezclaba cuestiones técnicas con humoradas y anécdotas. Vestía informal, remeras con consignas políticas o culturales, y estaba siempre disponible para "*ir por una cervezas*" luego de los encuentros en el *lab*.

En cuanto a los participantes, encontramos un perfil general en tanto que algunas diferencias. Su presencia era abrumadoramente masculina: sólo una de las catorce personas que permanecieron a lo largo del hackatón era mujer. En general se trataba

de estudiantes avanzados o programadores con alguna experiencia en desarrollo. Más allá de eso, todos se consideraban en mayor o menor medida usuarios de GNU/Linux.

Los participantes presentaban diferencias en cuanto a su apariencia. Algunos detentaban un perfil más "empresarial", con vestimenta y *look* de oficina, mientras otros a menudo se presentaban más informales, siguiendo un cierto estilo que puede identificarse con ciertos referentes del mundo de los programadores y *hackers*.

> *"Todos tenían sus laptops y en ellas diferentes distribuciones GNU/Linux que se dividían entre Debian, Ubuntu y Gentoo. En el aspecto diferían considerablemente. Christoph vestía una remera con la consigna "drop drums, not bombs". El resto de los participantes oscilaban entre un look informal, con barba, pelo largo, remeras grandes o ropa de algodón, buzos y joggins, hasta un look más vinculado con el trabajo de oficina o empresarial, con camisa arremangada, sweater y zapatos, bien afeitados y con peinados cortos. Uno de los participantes tenía una remera con el logo de Intel, una de las grandes firmas de software radicadas en la ciudad. Fue interpelado por uno de sus pares: 'también quiero una de esas!' ".*
> (Registro de campo, primer encuentro)

**Imagen 1:**
**Primer encuentro hackatón**

En cuanto a las edades también había variaciones. En su mayoría eran jóvenes de entre 20 y 30 años, pero había personas de mayor edad. Con el tiempo fui además descubriendo que los participantes se conocían entre sí. El vínculo entre ellos estaba marcado en gran medida por su pertenencia a la universidad. Algunos me comentaron que la carrera de informática no era muy masiva: "*como mucho se reciben 10 personas al año, por lo que casi todas las personas se conocen de antes, en alguna forma*."

**Los participantes: programadores, hackers, geeks.**

Como anticipamos en el primer apartado, mi lugar en la experiencia era el de observar "*geeks"*. Los participantes se auto-denominaron desde un comienzo como geeks. Ahora bien, ¿a qué se referían con esta expresión? Nos detendremos en este término así como el de *hacker*, otra denominación también comúnmente utilizada en la jerga de los informáticos.

Tal como lo definen algunos investigadores, la expresión *hacker* se refiere, en primer lugar, a personas que se dedican a "programar de forma entusiasta" y creen que "poner

en común la información constituye un extraordinario bien, y que además para ellos es un deber de naturaleza ética compartir su competencia y pericia elaborando software gratuito y facilitando el acceso a la información y a los recursos de computación siempre que ello sea posible" (Himanen, 2001).

El término surgió en los Estados Unidos entre un grupo de programadores del *Massachusetts Institute of Technology* (MIT) que empezaron a llamarse hackers a principios de la década de 1960. Con posterioridad, sin embargo, el significado del término fue distorsionado. A mediados de la década de 1980, diferentes medios de comunicación empezaron a hablar de hackers para referirse a los criminales informáticos (Raymond, 1992). A fin de evitar la confusión con aquellos que dedican su tiempo a escribir virus informáticos y a colarse en los sistemas de información, los hackers empezaron a denominar "crakers" a estos usuarios destructivos.

La actividad de estos informáticos entusiastas se acompaña a menudo de una "ética *hacker*" que les es propia (Levy, 1984; Himanen, 2001). En ella, el trabajo se emparenta con un compromiso por la libertad de la información, con la pasión por la programación, con el hecho de pertenecer a comunidades de pares, con la creatividad y el reconocimiento del mérito, al mismo tiempo que su actividad se refuerza como valorable por la sociedad. Coleman y Golub (2009) complejizan aún más esta definición asociando diferentes prácticas entre los *hackers* a un tensión entre el liberalismo y un *ethos* libertario.

El término *hacker* se refiere así a una forma particular de concebir el trabajo de programador. En relación con este, la expresión *geek* se refiere de manera más amplia a los modos de vida de estos sujetos fascinados por las tecnologías y la informática, así como a toda una "cultura", en tanto conjunto de elementos significativos, que los rodea. Ella ha sido retratada en algunas series televisivas, tales como *The IT Crowd,* en las que se muestra la cotidianidad de los informáticos y sus relaciones sociales. El gusto por la ciencia ficción, los juegos de rol, video-juegos y cómics, además de una pasión por la tecnología y el mundo de la informática son algunos de los puntos que definen al estereotipo *geek*. En el mismo sentido, existe una asociación entre las tecnologías libres y la "cultura hacker" o la "cultura geek".

Tal como veremos, hasta el propio creador del sistema Linux, Linus Torvalds, se describe a sí mismo como un *geek*:

> *"I was a nerd. Geek. From fairly early on. I didn't duct-tape my glasses together, but I might as well have, because I had all the other traits. Good at math, good at*

*physics, and with no social graces whatsoever. And this was before being a nerd was considered to be a good thing."*
(Torvalds and Diamond, 2001: 4)

El Jargon File, glosario popular entre los informáticos[v], coincide en este sentido en que el término geek estaba hasta hace poco tiempo asociado a connotaciones negativas o peyorativas[vi].

Retomando el caso analizado, encontramos situaciones diferentes en relación a ambas categorías. Algunos entre los participantes del hackatón auto-adscriban a estas definiciones como una reivindicación de su manera de entender el oficio de programador y el estilo de vida ligado a este. En especial Christoph era definido a menudo como *hacker*, indicando de alguna manera al mismo tiempo su jerarquía y experiencia.

Otros se sentían quizás interpelados de manera parcial por estos términos, en una suerte de posición de compromiso entre las connotaciones positivas y negativas asociadas con tales términos. De la misma manera, algunos tuvieron la oportunidad de marcar sus diferencias al respecto. Uno de los participantes me comentó que él "no era en realidad fanático de Linux, pero que en la facultad había varios que sí" lo eran. En la facultad era además conocida la práctica de algunos geeks que se reunían "los fines de semana a tomar cerveza y pasar toda la noche jugando video-juegos." (Registro de campo, tercer encuentro).

Algunos elementos que observamos a lo largo de la experiencia coinciden con el perfil que venimos señalando. Estos se relacionan con el uso y la reivindicación del uso de la computadora bajo *línea de comandos*, es decir, sin entorno gráfico, lo cual demostraba un diferencial de destreza al tiempo que ciertas reminiscencias hacia los albores de la era informática.

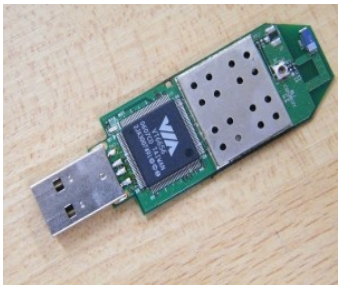


**Imagen 2: VT6656 USB WiFi Adapter**

El entusiasmo por la tecnología y los artefactos tecnológicos quedó también evidenciado en algunos registros del blog como el que presentamos a continuación:

*"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, 14/08/2009)

La “gloria” y admiración por los dispositivos electrónicos y su asociación -en clave de juego- con una suerte de libido hacia este tipo de objetos, cierran la descripción que venimos realizando.

**Programar el kernel Linux: desafío y diversión**

El sistema operativo GNU/Linux es un proyecto de grandes dimensiones que lleva 20 años de construcción progresiva y permanente. Desde sus primeras líneas de código escritas en 1991 por Linus Torvalds -mentor del emprendimiento hasta la actualidad y de quien tomaría su nombre- el *kernel* o núcleo del sistema operativo Linux se ha constituido en uno de los mayores proyectos de programación existentes, así como en una de las mayores experiencias de construcción colaborativa de software. Según el último informe disponible, el núcleo cuenta (en su versión 2.6.35) con unos 33.335 archivos y casi 13,5 millones de líneas de código. Su desarrollo continúa en la actualidad a un ritmo vertiginoso: entre 8 y 12 semanas es el tiempo promedio entre cada una de sus versiones estables, las cuales incorporan nuevas funcionalidades y soporte de *hardware* (Linux Foundation, 2010).

El *kernel* GNU/Linux toma por base los principios del movimiento software libre. Su código fuente se publica junto a las versiones ejecutables, de manera que está permanentemente disponible y accesible en el espacio de la red. Licencias libres del tipo Licencia Pública General GNU (GPL) protegen el código para que no sea apropiado sin ser luego liberado. El trabajo sobre el *kernel*, se nutre así de los aportes de miles de colaboradores alrededor del mundo. Raymond (2001) se ha referido a este en términos de un *modelo bazar,* por su dinámica horizontal y "bulliciosa" similar a un mercado de libre competencia.

Tuomi (2006) ha analizado el crecimiento este sistema como un proceso de co-evolución técnica y social. El crecimiento gradual de sus líneas de código ha ido acompañado de un número creciente de programadores vinculados con diferentes partes del proyecto. Su construcción cotidiana muestra así la tarea de coordinar la actividad de un gran número de personas con motivaciones diferentes, al tiempo que se va definiendo una suerte de contrato social entre los participantes. La estructura de la

red sirve para la resolución de conflictos, la toma de decisiones y la regulación y gestión del proyecto.

La arquitectura del sistema GNU/Linux presenta asimismo una elevada complejidad. Su división modular con interfaces que permiten su interacción, posibilita la articulación de una gran cantidad de elementos, al tiempo que sus desarrolladores pueden especializarse en pequeños tramos de código sin tomar en consideración la totalidad de la estructura (Tuomi, 2006). El *kernel* es, así, muchos proyectos al interior de un proyecto de mayor dimensión.

Los elementos mencionados hacen que la intervención sobre el código del *kernel*, aunque sea en una porción muy pequeña y marginal del mismo, se convierta en un gran desafío para muchos programadores. Se genera así una suerte de "mística" en torno a este sistema y sus desarrolladores, presente en las representaciones de quienes forman parte del mundo de la informática. El fragmento citado a continuación da cuenta de ello:

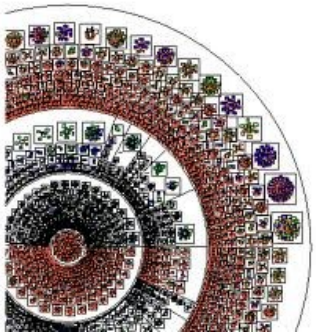


**Imagen 3: Arquitectura del kernel Linux**

> *"Por cierto, gente en Argentina que programa en el kernel ha habido desde hace ya bastante tiempo. (...) hasta yo mismo he contribuido mi propio parche. 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.*
>
> *Sin embargo, la leyenda dice que desarrollar para el kernel es reeeeedifíiiiiicil, 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, 14/08/2009)

La humildad con la que este programador se refiere a su propia contribución, la consideración de sus desarrolladores como "simples mortales" y del núcleo como "un programa como cualquier otro", asociado a las "leyendas" sobre el mismo y su dificultad para iniciarse en las destreza necesarias para comprenderlo, dan cuenta de algunos de los aspectos a los que nos referimos.

El desafío asociado a contribuir con el *kernel*, junto a los demás elementos que acabamos de comentar, dan cuenta de la “diversión” asociada a esta actividad. En el mundo de la programación, la diversión a menudo se relaciona con la realización de aquellas actividades más complejas, de difícil resolución, o que requieren de una mayor creatividad. Quienes se dedican a la programación, sea o no en el ámbito del software libre, a menudo presentan una notable “vocación” por su trabajo que se expresa en la pasión, la "satisfacción" y la "adrenalina" de construir código[vii] (Montes Cató, 2010: 81). Ello se evidencia en su dedicación y compromiso con los desarrollos de los que forman parte y, en otras ocasiones, en la dedicación de una parte de su tiempo libre a este tipo de tareas, realizadas a modo de hobby.

En el caso del hackatón, al desafío y la diversión se sumó la posibilidad de aprender e interactuar con otras personas que comparten el interés común por este tipo de tecnologías y con un *hacker* de una gran experiencia de programación sobre el núcleo. Debemos destacar, por último, la posibilidad de realizar un evento de este tipo en el medio local, en una región del interior de Argentina que, si bien presenta un crecimiento notable de sus sectores tecnológicos y de software (Berti y Zanotti, 2010), se encuentra aún lejos de los principales centros de desarrollo a nivel mundial. La distribución de desarrolladores del *kernel* da cuenta de esta tendencia.

**El desarrollo del hackatón**

Tal como anticipamos, el objetivo del curso sería la programación de un controlador GNU/Linux para un dispositivo de red inalámbrico portátil. Un controlador o *driver* es lo que permite que los equipos reconozcan sus diferentes piezas de *hardware* presentes y puedan utilizarlas. Al finalizar el taller, si se lograra conseguir exitosamente el objetivo planteado, el nuevo *driver* sería enviado a la *rama de desarrollo* del *kernel,* para ser incorporada en su próxima versión. Esto quiere decir que el producto realizado en el taller se transformaría en una parte constitutiva del núcleo de este sistema, y pasaría a estar disponible en cada una de las computadoras que lo utilicen.

El soporte para dispositivos en GNU/Linux suele ser un tema de controversias. Mientras en la actualidad el sistema ha crecido en usuarios y en soporte, hasta hace no muchos años los problemas con el *hardware* se contaban entre los principales obstáculos en la difusión hacia usuarios no expertos. Entre las empresas fabricantes, encontramos que mientras algunas ofrecen soporte para este sistema de manera habitual, otras se resisten por diferentes motivos a brindar los controladores requeridos.

En algunas ocasiones ellas optan por liberar sus especificaciones -un requisito fundamental para poder construir el controlador-, de manera que algún grupo de programadores puedan iniciar por su cuenta esta tarea. En otros casos, las empresas no dan soporte y tampoco liberan las especificaciones, con lo que los usuarios se ven imposibilitados de usar cierta parte de su *hardware* y los desarrolladores ven dificultado su desarrollo.

Aún más, en aquellos casos en los que los fabricantes sí ofrecen controladores para sus dispositivos, estos no siempre satisfacen a los usuarios y desarrolladores. Lo anterior puede deberse a que sus funcionalidades son limitadas o no igualan a las de otros sistemas operativos, o a que los controladores no cumplen con los principios del software libre, siendo su código privativo o cerrado. Esto ha conducido en varias oportunidades a que surjan intentos de reemplazar los *drivers* existentes por otros equivalentes, construidos comunitariamente con los requisitos de libertad del código abierto. La complejidad de estos esfuerzos se concreta en ejercicios de *ingeniería inversa*, procedimiento mediante el cual se *testea* el aparato en repetidas oportunidades intentando analizar las respuestas que presenta ante diferentes instrucciones, para de esta manera reconstruir cuáles son los conjuntos de instrucciones que determinan su funcionamiento.

En el caso del hackatón la situación se asemejaba en varios de los elementos planteados. Se trataba de adaptadores que habían sido recientemente lanzados al mercado, no contaban hasta el momento con soporte en este sistema y la empresa no tenía planeado desarrollar los controladores. Las especificaciones del mismo eran vagas y la documentación escasa. Lo único que hasta el momento podía ayudar a resolver la tarea era el código de un controlador anterior para otro sistema operativo que había sido "*porteado*" (adaptado) para funcionar en GNU/Linux, pero su desarrollo era experimental y sus funcionalidades escasas.

La tarea parecía así complicada y desafiante. Sin las especificaciones del dispositivo y sin soporte de parte de la empresa, se complicaba la tarea. A lo anterior se sumaba que al momento del primer encuentro los dispositivos donados por el fabricante aún no se encontraban en manos del equipo, debido a que no habían conseguido ingresar al país por cuestiones burocráticas.

Con los objetivos trazados, el primer encuentro se propuso establecer un cronograma de trabajo y empezar a poner en común algunos de los requerimientos básicos para realizar las tareas. Los encuentros quedaron fijados dos veces a la semana, durante el

lapso de un mes. Tendrían una duración de tres horas cada uno. En todas las sesiones se debería cumplir un avance hacia la construcción del *driver*, pasando por una sucesión de etapas técnicas determinadas. Las tareas que quedaran pendientes deberían terminarse en los días intermedios hasta el próximo encuentro.

El primer paso fue entonces asegurarse de que todos los participantes contaran con las herramientas de software necesarias para realizar el trabajo y estuvieran familiarizados con su uso. Enseguida comenzaron a hacerse evidentes los conocimientos previos de cada uno de los programadores, al tiempo que todos parecían encontrar elementos novedosos. Una entrada en el blog especifica sobre las herramientas utilizadas:

> *"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 que no cometamos cierto tipo de errores."*
>
> (VT6656 Linux Driver blog, 11/08/2009)

El trabajo con *git* permitía que los programadores pudieran ubicar e integrar las modificaciones realizadas por otros y por ellos mismos sobre el código. Las tareas se realizaban casi enteramente en una *consola terminal* y requerían bajar materiales de Internet por medio de un protocolo especial.

Los primeros encuentros tuvieron además un carácter más expositivo. Christoph mostraba como realizar algunas operaciones, las que se proyectaban en una pantalla central. Luego circulaba por la sala atendiendo consultas y los casos especiales que iban apareciendo. En algunos casos los participantes proponían soluciones.

**"Programando en libertad": Trabajo colaborativo y más.**

Luego de estos primeros pasos, el hackatón consistió fundamentalmente en una experiencia de programación colaborativa. ¿Cómo se desempeñaba esta tarea?

El trabajo de programación consiste básicamente en escribir fragmentos de código, de acuerdo con una serie de reglas correspondientes a diferentes lenguajes que se estén utilizando. En este caso se sumaban además las variables que ofrecía el *hardware* y el mismo *kernel*. Sobre un programa similar a un editor de textos simple, las diferentes partes del código aparecen en diferentes colores, para facilitar su rápida identificación.

Se trata de una tarea que requiere bastante atención a la coherencia entre las diferentes partes y su lógica interna.

En el caso que analizamos esta se trató además -y en una importante medida- de un ejercicio constante de ensayo y error. Los programadores se manejaban rápidamente sobre los fragmentos de código, borrando, escribiendo y modificando diferentes partes, en función de pruebas que realizaban sobre el dispositivo. La tarea consistía en ir agregando nuevas instrucciones e ir probándolas para verificar su funcionamiento, conectando y desconectando los adaptadores cada vez para 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.

Otra cuestión a destacar es que no existía una única forma de resolver los problemas con el código. Al igual que en otros ejercicios de escritura, existían diferencias de "estilo" y algunas soluciones solían mostrarse más “prolijas” o mejores que otras, ya sea porque proponían soluciones más simples, no interferían con otras partes del código, entre otros factores. La escritura revestía así un cierto carácter "artesanal", más allá de los procesos estrictamente técnicos que suponía. En el código se incluían además mensajes para los programadores que indicaban el uso de determinados fragmentos de manera de simplificar su lectura.

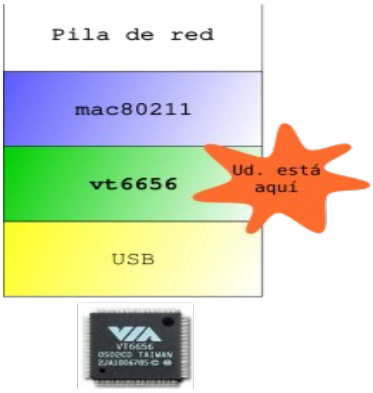



**Imagen 4: Etapas sucesivas del desarrollo**

En la medida en que las soluciones posibles eran múltiples, las versiones de código que manejaban los programadores necesitaban cada tanto 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 inteligente” y enviárselas para que fueran incorporadas.

El hackatón fue avanzando encuentro tras encuentro. Los participantes se mantuvieron hasta el final pese a las sucesivas complicaciones que la tarea fue presentando, las cuales pudieron ser sorteadas de diferentes modos. Estas terminaron por requerir bastante más tiempo del pautado originalmente: algunos dedicaban varias horas de trabajo los fines de semana, “*hasta las dos de la mañana*”, para intentar reparar fragmentos de código defectuosos. Junto con las complicaciones aumentaba el desafío y el entusiasmo, aunque la complejidad de la tarea podía conllevar algún nivel

de frustración.

Tal como propusimos al comienzo, el incentivo de un hackatón es el hecho de “programar en libertad”, esto es, hacerlo voluntariamente y en circunstancias agradables y elegidas por los participantes, sin presiones ni condicionamientos exteriores. Esto se observaba en los encuentros. En algunas ocasiones los programadores proponían apagar las luces del *lab* para mejorar su concentración y quedaban así en la penumbra, iluminados únicamente por las pantallas de las laptops, el reflejo del proyector y una luz colateral que ingresaba por un costado de la sala.

El trabajo implicó una serie de instancias de colaboración, así como la adquisición de nuevas destrezas y habilidades. El intercambio entre los programadores se hacía más fluido a medida que avanzaba la experiencia: estos compartían soluciones y recomendaciones, así como algunos pedazos de código que parecían resultar satisfactorios. Al mismo tiempo, cada uno parecía celoso de su propio trabajo. Se suscitaba así una suerte de competencia -más en clave de juego- por ver quién lograba resolver ciertas situaciones en el menor tiempo, planteando una resolución exitosa. Christoph asesoraba a cada uno que lo necesitaba y animaba a los que lograban resolver una parte, a seguir adelante con las tareas pendientes.

La dinámica de trabajo cambió en las últimas reuniones. Del trabajo integrado entre la totalidad del equipo, se pasó a una modalidad de grupos. Los programadores decidieron dividirse las tareas remanentes y trabajar por partes. A medida que los grupos se tornaban más reducidos y el trabajo más intensivo, comencé a sentir que interfería en las tareas. Me trasladé así a la sala contigua para intercambiar con los organizadores de la maratón. A medida que los equipos terminaban con su parte, debían esperar al resto para continuar. Los programadores aprovechaban ese tiempo para explorar algunos otros aspectos del dispositivo y experimentar con algunas funcionalidades desconocidas.

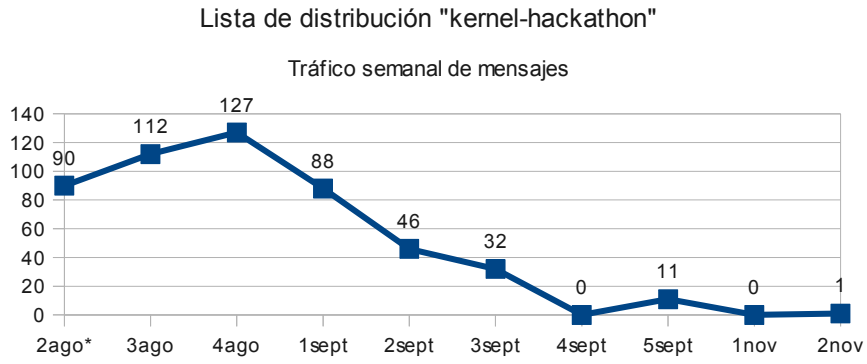
El trabajo se había ido intensificando hacia la última semana del taller. Aunque algunos equipos habían logrado su cometido, quedaban aún varias tareas pendientes. La parte más difícil había resultado la *transmisión*, la cual, junto con la *recepción de señales*, terminarían por permitir que el dispositivo funcionara. Ambas resultaron ser más complejas de lo esperado y llevarían un poco más de tiempo. El equipo probó un par de soluciones y se discutieron posibles lineas a seguir. Era claro de que no estaría listo para el final del hackatón. Los organizadores propusieron entonces tomarse una semana más y realizar, para concluir, una presentación abierta al público en la que se

socializara cómo había transcurrido el proceso.

**Listas de distribución: interacciones intensivas y regladas**

Los intercambios entre los participantes del hackatón se daban tanto en situaciones cara a cara como mediadas. La principal vía de comunicación en este último sentido era la lista de distribución “kernel-hackaton”, creada para tal fin por la comunidad de software libre local y albergada en sus propios servidores. A poco de comenzada la maratón, los organizadores me permitieron sumarme a la lista. A partir de allí pude analizar cómo fluía la comunicación entre los participantes. En los primeros días recibí cerca de 100 correos de la lista, lo que daba cuenta de la fluidez de la relación creada por este medio. Aunque en su mayoría estos se referían a cuestiones técnicas, otros tenían por finalidad organizar salidas o mostrar la difusión que se iba haciendo del evento. Un análisis pormenorizado de los mismos trasciende sin embargo los límites de esta presentación.

El gráfico siguiente muestra la intensidad de las comunicaciones a lo largo y después de finalizado el taller:




* el primer valor de la serie es estimativo.

Fuente: elaboración propia.

Como se observa, la intensidad de los intercambios fue creciente a lo largo de los encuentros hasta el final del hackatón, en la última semana de agosto. Luego de ello encontramos un periodo de distensión durante las semanas subsiguientes, a medida que se iban resolviendo cuestiones que habían quedado pendientes en el desarrollo del *driver*. Aunque no se encuentra representado aquí, un periodo de más larga duración da cuenta además de sucesivas instancias de contacto esporádico entre los integrantes.

La participación en listas de distribución supone el conocimiento de una serie de pautas que regulan las interacciones entre sus suscriptores. El ejercicio del hackatón

sirvió en este sentido para que los participantes aún no familiarizados aprendieran, como en una suerte de ejercicio preparatorio, a formar parte de listas de desarrollo habituales entre los programadores del *kernel*. Tal como indicaba una entrada del blog, es importante además de aprender a programar, "aprender a comportarse como un buen desarrollador". Ello implica el seguimiento de ciertas reglas que se presentaban al detalle:

> *"(...) las reglas de "buena educación" en algunos proyectos de software libre pueden ser bastante diferentes de las del mundo real. Por ejemplo, si bien es importante tratar de ser cordial y respetuoso de los demás, probablemente sea más importante el desarrollar la capacidad de no ofenderse cuando otros no lo son: en listas muy técnicas, como las del kernel, es bastante común que algunas personas se comporten de manera descortés, y sobre todo que critiquen el trabajo de otros de manera muy agresiva. Lo peor que uno puede hacer en estos casos es ofenderse y responder a la agresión, lo mejor es tomárselo con calma, quizás incluso tomarse un día entero para calmarse, y recién después contestar a los puntos técnicos, dejando de lado la potencial ofensa personal.*
>
> *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, 14/08/2009)

No ser descortés, no alterarse ante las críticas despiadadas, responder con argumentos técnicos en lugar de ofensas, así como preparar "parches" correctamente, son algunas de las reglas fundamentales. Los "parches" ("patchs" en inglés), son archivos con cambios, en general pequeños, que se aplican a un programa o su código fuente para corregir errores, agregar funcionalidades, actualizarlo, entre otras finalidades. Los parches deben cumplir a su vez con las siguientes pautas: 1. deben ser la solución completa a un problema; 2. no se debe enviar más de un parche por cada correo; 3. el asunto del mensaje debe indicar que este contiene un "[PATCH]"; 4. los parches deben contener una linea "Signed-off-by:" indicando quien es su autor; 5. antes de enviarlo se debe revisar que respete las convenciones de codificación del *kernel*. (VT6656 Linux Driver blog, 14/08/2009)

Estas prerrogativas y modos de comportamiento dan pistas acerca del funcionamiento de las comunidades de software libre, las cuales se basan intensivamente en este tipo de recursos para coordinar las tareas de programación.

***Más allá de la programación: salidas, socialización, cervezas.***

Tal como lo hemos comentado, el proceso de desarrollo consistió en una tarea compleja, que requirió de una importante cantidad de tiempo y dedicación por parte de los participantes. Pero no todo fue trabajo. El hackatón consistió también en un espacio de socialización e incluyó momentos de diversión. La salida "por unas cervezas" se convirtió así en una suerte de rutina posterior a los encuentros, a lo que también se sumaron diferentes reuniones durante los fines de semana. La alusión a la cerveza aparecía constantemente: *"Te explico si me pagas unas cervezas*", envases de cerveza en el *lab*, entre otras, apoyando una tradición de alto consumo de este tipo de bebida que me fue referida en varias oportunidades.

Algunos trabajos similares sobre comunidades de programadores dan cuenta, en un mismo sentido, de esta combinación entre trabajo intensivo, socialización y diversión que caracterizan a este tipo de encuentros. El hecho mismo de la congregación cara a cara entre personas que comparten ciertos componentes identitarios e intereses en común redunda en un encantamiento de la experiencia:

> "...las conferencias de hackers son rituales de confirmación, liberación, celebración y en especial re-encantamiento, 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 celebración condensatoria, los hackers imbuyen sus acciones de nuevos significados, revitalizados o éticamente recargados. (...) Se trata de profundos momentos de re-encantamiento cultural, en donde los participantes construyen y comparten una experiencia engrandecida de sí mismos.
> (Coleman, 2010:53 [traducción propia])

Desde la organización se alentaban así este tipo de prácticas. Se buscaba que los participantes compartieran tiempo con Christoph para hacerle conocer la cuidad y ponerlo al tanto de algunas costumbres locales. Los avances sobre el proyecto también constituían a menudo un motivo de celebración. De esta manera se iba creando un vínculo que trascendía ampliamente las actividades profesionales.

**El balance de la experiencia desde la visión de sus participantes.**

> *"Queridas gentes,*
> *ya han pasado varios meses desde el final del taller, y por lo que alcanzo a ver, si bien se ha cumplido el objetivo docente de enseñarles técnicas de programación en el kernel, y se cumplió parcialmente el objetivo técnico de hacer un driver para vt6656, no se ha cumplido el objetivo que nos habíamos propuesto desde Vía Libre: motivar a un grupo de desarrolladores a continuar el proyecto más allá de la vida del curso, para que ese grupo a su vez sirviera de motivador para otros. (...)"*

(“kernel-hackatón” mailing list)

El fragmento citado resume en pocas palabras el balance que los organizadores realizaron de la experiencia. El taller logró completarse en su totalidad y se logró avanzar significativamente en el armado del controlador para el adaptador inalámbrico *vt6656*. Ello implicó sortear una serie de dificultades que se fueron suscitando. Más allá de estos logros -trascendentes desde el punto de vista de que no existían experiencias de este tipo realizadas localmente- no se logró sin embargo consolidar un equipo de trabajo duradero que fuera capaz de continuar el trabajo e incluso multiplicar la experiencia. La evaluación fue así no del todo satisfactoria.

En un nivel más amplio, el informe realizado luego de concluida la maratón (FLOSSWorld, 2010) da cuenta de una baja participación en proyectos de software libre en Argentina y en la región en general. La propuesta buscó en este sentido generar un “*efecto demostración*” que diera cuenta de que es posible concretar proyectos de alto perfil sobre este tipo de tecnologías en el medio local, superando la falta de confianza y el “miedo” a las dificultades del *kernel* que analizamos con anterioridad.

El hackatón dejó a su vez al descubierto algunas “*barreras no anticipadas*” que debieran ser tenidas en cuenta para futuros proyectos. Entre los obstáculos se mencionan: dificultades en el acceso a fabricantes, el escaso interés de los mismos en colaborar con el desarrollo, la escasez de contactos personales para facilitar ambas tareas anteriores, y la necesidad de dominio de la lengua inglesa, considerada un imprescindible dentro del mundo informático. Al parecer, el contacto y la colaboración con fabricantes no constituyen aún una práctica muy difundida en el medio local (FLOSSWorld, 2010).

Más allá de estas dificultades, los participantes mencionaron otros aspectos que, en su conjunto, impidieron la continuidad del proyecto más allá de los límites del maratón. Entre ellos se destacan la falta de tiempo libre para dedicarle al proyecto, la falta de una persona que asuma el liderazgo del equipo y coordine las tareas, y la falta de interés individual en el producto final, es decir en el propio controlador (FLOSSWorld, 2010). Sus respuestas dan más pistas sobre las motivaciones presentes detrás de este tipo de desarrollos.

**Cierre: “no sólo del código vive el hacker[viii]”**

A lo largo de la presentación buscamos reconstruir la experiencia del hackatón y

resaltar algunas de las dimensiones que encontramos más significativas a partir del abordaje realizado. Como pudimos observar, más allá de los múltiples aspectos técnicos que rodean a este tipo de actividades, encontramos al mismo tiempo un sinnúmero de representaciones y prácticas en torno a los mismas.

La auto-percepción como hackers o como geeks y las disputas y reivindicaciones en torno a estos términos, las leyendas románticas que acompañan y generan una suerte de "mística" alrededor del *kernel* GNU/Linux y sus desarrolladores, la pasión por el desarrollo de tecnologías y el desafío de arremeter contra tareas complejas, el reconocimiento de la experiencia, el mérito y la competencia que conllevan los episodios de trabajo colaborativo, la multiplicidad de motivaciones que se promueven entre los miembros de comunidades de software libre y los valores que se vehiculizan y refuerzan, las pautas de "buena educación" y los modos de hacer que orientan los intercambios entre programadores, la intensidad de las experiencias de co-presencia tanto en términos de trabajo como de socialización y diversión, así como la "militancia" que implica para muchos representantes de estas comunidades y organizaciones la proliferación de los espacios de participación y creación colectiva, son sólo algunos de los elementos que componen el mundo cotidiano de estos programadores, entusiastas informáticos.

**Bibliografía:**


BERTI Natalia y ZANOTTI Agustín (enviado) "Nuevas Industrias: políticas públicas y gobernanza en la industria del software y servicios informáticos. El caso de Córdoba, Argentina". En: *Trabajo y Sociedad,* ISSN 1514-6871.

COLEMAN Gabriella (2010) "The Hacker Conference: A Ritual Condensation and Celebration of a Lifeworld". En: *Anthropological Quarterly*, Vol. 83, No. 1, pp. 47–72, ISSN 0003-549.

COLEMAN G. y GOLUB A. (2009) "Hacker practice . Moral genres and the cultural articulation of liberalism" . *Anthropological Theory* 2008; 8; 255 .

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

HIMANEM Pekka (2001) *La ética hacker y el espíritu de la era de la información*. Disponible en: www.geocities.com /pekkahacker/ [Citado: 14/8/2009]

LEVY Steven (1984) *Hackers: Heroes of the Computer Revolution.* Delta, New York.

LINUX FOUNDATION (2010) *Linux Kernel Development. How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It: December 2010*. KROAH-HARTMAN Greg, CORBET Jonathan, MCPHERSON Amanda. Disponible en: www.linuxfoundation.org/sites/main/files/publications/whowriteslinux.pdf [Citado: 24/8/2010]

MONTES CATÓ J. S. [coord.] (2010) *El trabajo en el Capitalismo Informacional. Los trabajadores de la industria del software* . Poder y trabajo editores, Benavídez.

RAYMOND Eric (1992) *Una breve historia de los hackers*. Alberto Pintado Sánchez (Trad.) O'Reilly. Disponible en: usuarios.multimania.es/apintado/BHHackerdom/ [Citado: 6/07/2011]

RAYMOND Eric (2001) *The cathedral & the bazzar. Musings on Linux and Open Source by an accidental revolutionary*. Ediciones varias. Disponible en: es.tldp.org/Otros/catedral-bazar/cathedral-es-paper.html [Citado: 6/08/2010]

TORVALDS Linus and DIAMOND David (2001) *Just for Fun: The Story of an Accidental Revolutionary*. HarperCollins Pub, US.

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

**Sitios web consultados:**

*The Jargon File: Geek*. <http://www.catb.org/jargon/html/G/geek.html> [Citado: 14/07/2011]

*VT6656 Linux Driver. Developing a Linux Driver for VIA's VT6656 USB WiFi Adapter*

<http://vt6656.wordpress.com>

*"kernel-hackathon" mailing list.*

<http://mx.grulic.org.ar/lurker/list/kernel-hackathon.en.html>

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

*Linux, Wikipedia The free encyclopedia* <en.wikipedia.org/wiki/Linux>

**Entrevistas Citadas:**

*Entrevista a Hellwig*. 26 de agosto de 2009.

i Nos referimos a la familia de sistemas operativos que usan el núcleo o *kernel* Linux. Este permite el funcionamiento de una amplia variedad de dispositivos electrónicos, desde teléfonos móviles, tablet PCs, enrutadores, consolas de video-juegos, hasta computadoras de escritorio servidores y superc-omputadoras. Linux es el sistema más utilizado en servidores y corre las 10 super-computadoras más poderosas en el mundo. (Adaptado y traducido de Wikipedia)

ii El modelo propietario concibe al software como un sector más de la economía. Su producción se basa en la creación de bienes informacionales y su consideración como bienes tradicionales, regidos bajo el criterio de la escasez bajo la regulación del mercado en términos de oferta y demanda. El modelo propietario basa sus ingresos en la comercialización de las diferentes piezas de software, protegidos por licencias o patentes que restringen su copia, distribución, modificación y uso. Se distribuye en formato binario o ejecutable unicamente, lo que equivale a decir que su código es cerrado y mantenido fuera del dominio público.

iii El software se presenta en dos formas básicas: como código ejecutable y código fuente. El código ejecutable es aquel que se encuentra listo para ser utilizado como una herramienta o aplicación. Ha sido compilado como código binario, esto es, un conjunto de ceros (0) y unos (1) capaces de ser procesados por los dispositivos electrónicos. El código fuente, por su parte, contiene las instrucciones escritas en lenguaje de programación que el mismo lleva a cabo para lograr las tareas que se propone. Es, en definitiva, el “texto original” del programa, el que permite su inteligibilidad, su aprendizaje y su intervención para elaborar futuras piezas de software o realizar modificaciones sobre las ya existentes..

iv VT6656 Linux Driver blog, 18/08/2009.

v “Los primeros artefactos intencionales de la cultura hacker - las primeras listas de argot, las primeras sátiras, las primeras tímidas discusiones de la ética hacker todo ello fue propagado en ARPANET en sus primeros años. En particular, la primera versión del ``Jargon File" <http://www.tuxedo.org/jargon> fue desarrollada a través de una intrincada colaboración durante 1973-1975. Este diccionario de argot se convirtió en uno de los documentos definitorios de la cultura.” (Raymond, 1992)

vi “Antes de los años 1990 el uso del término era más bien negativo. Las versiones anteriores del léxico definen al geek de la computadora como aquel que se alimenta de errores en el código como forma de vida -un asocial, maloliente, cara pálida, monomaníaco con toda la personalidad de un rallador de queso. Esta a menudo sigue siendo la manera en que los geeks son considerados por los no-geeks. Pero, en la medida que la cultura dominante se vuelve más dependiente de la tecnología y las habilidades técnicas, las actitudes tradicionales han tendido a desplazarse hacia un respeto a regañadientes. En consecuencia, hay ahora festivales del 'orgullo geek '(la referencia implícita al 'orgullo gay' no es casual).”
(Jargon File, 2011 {traducción propia})

vii Cabe mencionar que el trabajo compilado por Montes Cató (2010) incorpora estos elementos en relación con modos de trabajo precarizados y flexibles en varios nichos de la producción de software en Argentina.

viii VT6656 Linux Driver blog, 14/08/2009