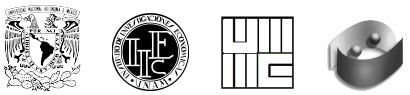
Construcción Colaborativa del Conocimiento

El presente trabajo forma parte del libro que recoge los trabajos del Seminario Construcción Colaborativa del Conocimiento (ISBN: 978-607-02-2373-0), coordinado por Gunnar Wolf y Alejandro Miranda.
Puede encontrar el libro completo para su descarga, así como los demás capítulos de forma individual, en
http://seminario.edusol.info/seco3



Capítulo 4

Factores de motivación y elementos de reconocimiento

Gunnar Wolf

Mucha gente se ha preguntado qué lleva al crecimiento de las comunidades de desarrollo de conocimiento. Aquí nos centramos en dos cuestiones sobre los grupos relacionados con el software libre, así como su vinculación mutua, que es mayor de lo que podría parecer a primera vista:

1. Qué genera la identificación, la pertenencia de individuos a un colectivo o comunidad definida por un conjunto de *principios* (que con frecuencia no están definidos muy rigurosamente), y los motiva a participar activamente de diversas maneras.

2. Cómo se explica la aparente contradicción económica que lleva a miles de personas a coordinar sus esfuerzos para crear productos de alta especialización intelectual, sin que medie una retribución económica por este trabajo. Si hablamos de motivaciones más allá de los factores económicos, ¿qué motiva esa inversión de trabajo especializado?

Los grupos de desarrollo de software libre han diseñado diversas y complejas estructuras sociales, en su mayoría centradas alrededor

de la economía del regalo, tema que abordaremos más adelante, y con interesantes esquemas que llevan al posicionamiento de los individuos en una jerarquía.

Muchos de los puntos aquí mencionados también son aplicables a las otras modalidades de comunidades *basadas en* u *orientadas a* la generación colaborativa del conocimiento. Sin embargo, nos enfocaremos en la naturaleza de las relaciones en las comunidades del movimiento del software libre, con el propósito de acotar ciertas características sociales necesarias para tocar el tema de pertenencia e identificación.

En el presente trabajo, más que cubrir o comparar los diversos factores motivadores y los papeles de cada uno de ellos hacia el interior de las comunidades, abordaré algunas características emergentes y no obvias de las comunidades de desarrollo.

4.1 FACTORES DE MOTIVACIÓN

Las diferentes comunidades de desarrollo de software libre, e incluso ciertos proyectos específicos de desarrollo,[1] presentan distintos rasgos, más allá de los meramente técnicos, que determinan las características de la personalidad de sus participantes. A esas características las denominaremos *factores de motivación*: llevan a la participación de individuos con distintos perfiles, aunque con una cierta consistencia social. No sólo las comunidades como agregadores, sino también todo desarrollador como individuo, tienen diversos motivadores para involucrarse en proyectos libres.

Los factores motivadores que expondremos a continuación suelen no presentarse aislados. Para cada comunidad, cada proyecto y

[1] Una comunidad de desarrollo puede estar limitada a un solo proyecto, en especial cuando tiene un propósito técnico especializado y bien definido. Sin embargo, es muy frecuente que una comunidad sea suficientemente amplia, o muestre una diversidad de ámbitos de aplicación o subáreas de desarrollo, para que su trabajo y su total de desarrolladores abran un abanico de proyectos.

cada desarrollador, invariablemente habrá componentes –en menor o mayor grado– de todas las modalidades expuestas a continuación.

4.1.1 La programación como un acto de belleza o de arte

Programar va mucho más allá de plasmar mecánicamente ideas en un lenguaje formal. Los programadores con frecuencia encuentran un sentido *estético* ante el desarrollo de software, y es común que en el transcurso del desarrollo de un sistema, un buen programador invierta más tiempo en buscar la manera más elegante de implementar una solución que en ponerla en práctica.

Y si bien por tradición defenderemos la inversión extra de tiempo para lograr esta elegancia y hallar la solución general a un problema que pueda ser reutilizada en otros casos, la satisfacción intrínseca de hacer *algo bello* es suficiente motivador para un desarrollador que se apasiona por su profesión.

La manera en que un programador enfrenta un reto cognitivo va más allá de la resolución misma del problema. Es común encontrar profundas discusiones respecto a las *metodologías* o *escuelas de pensamiento* más adecuadas para expresar más limpiamente o con mayor elegancia un proceso lógico. Tal vez una de las mejores expresiones de esto sea el *Tao de la programación* (James, 1986). Mediante parábolas que imitan el estilo de Confucio, Geoffrey James expone varios principios de metodología, diseño de sistemas y administración de proyectos.

El acto de programar lleva al desarrollador, con cierta frecuencia, a una genuina sensación de contacto con lo divino: la computadora es un campo virgen en el cual, literalmente, nos es posible crear algo de la nada. Al programar, el desarrollador puede sentir una auténtica libertad, ante posibilidades infinitas, del mismo modo que lo experimenta un pintor frente a un lienzo en blanco: aun si fuese una creación utilitaria, por encargo, su ejecución brinda una gran libertad creadora.

Muchos desarrolladores vislumbran retos en especial jugosos al reimplementar *elegantemente* un problema antes resuelto de manera *sucia* o incompleta, y al analizar diferentes proyectos que se enfocan en un mismo nicho es posible observar incontables ejemplos de ello.

La apreciación (y por supuesto, la creación) estética del código, además, suele relacionarse con una subcultura *hacker* (véase más adelante). Como un último punto en esta dirección, me permito referir a lo mencionado en el tema "El software libre en tanto movimiento social", del apartado "Software libre y construcción democrática de la sociedad": una de las principales motivaciones de la militancia política de los *hackers* es la falta de elegancia y falta de lógica en la normatividad legal, y una compulsión por demostrar que esa normatividad, al ser parte fundamental del pacto social,[2] sencillamente resulta inadmisible que tenga semejantes carencias lógicas; ejemplos de ello (descritos en el tema citado) incluyen demostrar que el código puede ser visto y apreciado como arte y, por tanto, debe gozar de la protección de la libertad de expresión.

En este sentido, el *hacker*[3] no limita la apreciación de la estética, la elegancia y la coherencia lógica y funcional al código orientado a ser ejecutado por una computadora, sino que la aplica a los esquemas legales y sociales. Esto aclara la existencia, permanencia, reconocimiento y fuerza de organizaciones tan disímiles como el Chaos Computer Club (Wikipedia, 2008a) o la Electronic Frontier Foundation (Electronic Frontier Foundation, 2008).

4.1.2 Programar por diversión

Es innegable que un importante elemento de motivación es el gusto por lo que uno hace, por enfrentar un problema como un reto intelectual, no únicamente como una tarea que debe ser cumplida. Así

[2] El lenguaje del programador y el de los legisladores no es tan diferente; la diferencia radica en el sujeto que debe ejecutar dicho código.

[3] O como muchas veces se definen en este ámbito, el *hacktivista*.

lo ilustran Linus Torvalds y David Diamond (Torvalds y Diamond, 2001) desde el mismo nombre de su obra.

Si bien es frecuente señalar en ese punto, incluso muchas veces como una exageración propagandística, llega a formar parte importante de una motivación multifactorial. Dentro de las motivaciones que llevan a un individuo a programar y publicar software libre encontramos (Bitzer, Schröder y Schrettl, 2006):

1. La necesidad de una solución de software específico.
2. La diversión del reto, es decir, una recompensa de tipo *homo ludens*.
3. El deseo de dar un regalo a la comunidad programadora, es decir, un regalo-beneficio (véase en este capítulo la sección 4.1.5, "La economía del regalo").

   [...]

   El programar es visto como una actividad que se realiza en el tiempo libre, es decir, jugar con las posibilidades del software o conquistar un reto como un pasatiempo.

   [...]

   La idea de que la diversión proveniente del juego es un importante factor motivador para los humanos no es nueva, y puede ser rastreada hasta Platón; el *locus classicus* es Johan Huizinga (1938). El *homo ludens* de Huizinga, el humano juguetón, significa en nuestro campo que el programador recibe una cierta forma de beneficio sencillamente como resultado de programar o de vencer a un problema de software.

La motivación basada en la diversión va claramente relacionada con la orientada a la creación de belleza, discutida antes en esta obra. Podemos averiguar cuál es el factor determinante para un desarrollador (o un proyecto) específico enfocándonos en analizar

cómo está ejecutado: ¿parece ser más importante solucionar un problema o encontrar una solución elegante?

Este factor, sin embargo, es peligroso cuando predomina demasiado. Muchos proyectos han quedado abandonados o *huérfanos*, al dejar de ser suficientemente atractivos para mantener el interés de los desarrolladores, y muchos proyectos han puestos mayor importanca en resolver el problema que en hacerlo bien, llevando a código frágil y abriendo el sistema a vulnerabilidades.

Con todo, sea por la diversidad de actores o por las características de la personalidad de los autores del software libre, estos problemas tienden a presentarse incluso en menor grado que en el mundo del software propietario.

4.1.3 Semillero de interacciones sociales

Contrario a la percepción cultural y en congruencia con el análisis en este capítulo, los *hackers* en general no son ermitaños que evitan el contacto social. Por el contrario, tienden a formar sociedades con intercambios intensivos y multidimensionales. Escribir e intercambiar código es quizá el primer elemento de motivación, pero a éste pronto le siguen otros de identificación política y los elementos lúdicos con alta carga intelectual.

Los grupos de desarrolladores de software libre crean verdaderas comunidades, en el sentido más amplio posible. Incluso podría argumentarse acerca de si hay una correlación entre el grado de compromiso político-ideológico de una comunidad y su cohesión (y tal vez impermeabilidad).[4]

[4]Con base en observaciones empíricas y un universo demasiado limitado como para generalizar, encontramos que comunidades con una fuerte identificación ideológica, aun si carecen de un lenguaje técnico en común (es decir, hay decenas de lenguajes de computadora que se utilizan para su operación), como Debian, tienden a formar vínculos sociales muy cercanos y duraderos, y una relativamente alta proporción de sus integrantes viaja cada año al congreso DebConf, celebrado en rincones muy diversos del mundo, pero tienen

Ahora, que los *hackers*, creadores de software libre, sean sociables en el entorno propicio no significa que les sea fácil descubrir este entorno. Es muy común escuchar historias de desadaptados sociales durante la infancia, convertidos después en parte central de sus comunidades. El intercambio social cara a cara es inmensamente enriquecedor y, sin lugar a dudas, un factor de motivación central para muchos. Citando a Coleman (2010):

> Es cierto que los *hackers* no consideran este tipo de interacción diaria y en persona entre amigos y compañeros de trabajo como el *locus* de la "comunidad" típica cuando se refieren al *hackeo* o al SL/SFA.[5] Para muchos *hackers*, el *locus* de la socialización ocurre, como lo relata buena parte de la literatura, sobre la red y es translocal. Compuesto de un vasto y disperso conglomerado de gente –amigos cercanos, conocidos, extraños–, se ven a sí mismos unidos por un ferviente interés y compromiso hacia la tecnología, y conectados a través de las aplicaciones de internet que les permiten comunicarse y construir tecnologías.
>
> Sin embargo, si los *hackers* han logrado sin duda situarse a sí mismos en una amplia red local de comunicaciones, e imaginarse en términos de redes y virtualidad, lo han hecho cada vez más celebrando esta translocalidad en persona. Más que nunca antes, los *hackers* participan y disponen de un espacio físico común con muchos tipos de grupos sociales (como acadé-

complejos y largos ritos de iniciación por los que un interesado debe pasar para convertirse en miembro. Comunidades con una mayor homogeneidad lingüística y mayor tolerancia ideológica, como la de desarrollo del lenguaje perl (o más aún, la hoy muy floreciente y dinámica comunidad de Ruby on Rails), parecerían llevar una mucho mayor vida social, al celebrar tres o cuatro congresos anuales, pero la comunidad que asiste a cada uno de estos congresos es mucho menos constante y hay mayor proporción de participantes por vez única; del mismo modo, formar parte de estas comunidades es más sencillo, ya que por lo general basta con seguir una serie de instrucciones para recibir acceso y, por tanto, membresía.

[5] Software libre y software de fuente abierta.

micos, profesionales, aficionados, activistas y consumidores): el congreso [...].[6]

4.1.4 Enfatizando en las ventajas del modelo distribuido, descentralizado de desarrollo

En el texto *La catedral y el bazar* (CatB) (Raymond, 1997) se busca explicar desde un punto de vista antropológico al movimiento del software libre. Ese texto, que sentó las bases para la creación de la Open Source Initiative (Open Source Initiative, 1998), despertó un debate respecto al qué y cómo se organizan las comunidades de desarrolladores; se volvió casi de inmediato una referencia obligada respecto a la cual tanto los participantes del movimiento como los proyectos se catalogaron o tomaron posición.

Con el paso del tiempo, diversos proyectos comenzaron a ser catalogados en relación con ese escrito. Se entablaron innumerables discusiones respecto a si seguían un modelo *más catedral*, en el que un desarrollador o un grupo de ellos ejerce control estricto sobre la evolución del proyecto; o *más bazar*, en que el desarrollo se convierte más en la colaboración de grandes cantidades de individuos u organizaciones que trabajan cada cual por su lado, con fines quizá divergentes. A fin de cuentas, resulta casi imposible encontrar proyectos desarrollados con base en una *metodología bazar* pura.

No ocurre así por el lado tradicional, en las llamadas *catedrales*. La totalidad de los desarrollos propietarios, al igual que una gran cantidad de desarrollos libres pequeños (aquellos que son impulsados directamente por su autor primario y no han llamado la atención de suficientes colaboradores para dar el salto a un modelo más distribuído), así como los principales proyectos que en principio fueron propietarios pero después fueron *liberados* por sus empresas, se observan al día de hoy como *catedrales* tradicionales.

[6] En el original, la autora indica *conference*, que puede referirse desde a un congreso altamente técnico hasta a una exposición comercial.

Los principales ejemplos son MySQL, OpenOfice, Mozilla y OpenSolaris.[7]

Además de ésos, claro está, hay grandes proyectos cuya principal labor es de integración, de facilitar al usuario final el acceso al software, como las *distribuciones* de Linux.[8] Por la manera en que las distribuciones se relacionan con los miles de proyectos independientes que las forman, podemos hablar de *bazares de catedrales* en el caso de las distribuciones formadas mediante procesos comunitarios; o incluso de *catedrales de catedrales*, en el caso de aquellas dirigidas por una visión única, de una empresa.

Ahora bien, el CatB destaca las ventajas organizacionales, técnicas y económicas, principalmente motivado por las observaciones respecto al desarrollo distribuido y descentralizado como un *modelo de desarrollo de sistemas*. Si bien ha servido para explicar

[7]Cabe mencionar que, a excepción de Mozilla, todos los ejemplos citados son proyectos desarrollados o adquiridos por Sun Microsystems, empresa que, a lo largo del último decenio, luchó fuertemente para verse como una empresa interesada en aculturarse a y ser aceptada por la comunidad con fuerte carga ideológica del software libre. Sin embargo, en abril de 2009, Sun Microsystems fue adquirida por Oracle, una empresa que nunca ha destacado por su interés hacia la libertad del código y hay muchas columnas y agitadas discusiones en las que se especula respecto al futuro de estos proyectos. Varios meses más tarde y ya al momento de entregar la versión definitiva de este texto, aún quedaban amplias dudas acerca de cuál sería el futuro de dichos proyectos bajo la tutela de Oracle. La importancia estratégica y económica de dichos proyectos llegó incluso al grado de que, al seno de la Unión Europea, entidades regulatorias tuvieron que debatir acerca de si permitir esta adquisición, en virtud de que Oracle quedaría en posición de ejercer prácticas monopólicas al ser titular de los derechos de la popular base de datos libre MySQL.

[8]Una *distribución* es un conjunto de software suficiente para la operación básica de una computadora, mínimamente consistente del núcleo del sistema operativo y los *paquetes* necesarios para tener un entorno Unix operable, aunque típicamente con todos los programas que un usuario general requerirá; hoy en día hay cientos de distribuciones, desde las más mínimas, ocupando pocos *megabytes*, hasta las más completas, con decenas de *gigabytes* y decenas de miles de paquetes independientes.

las innegables ventajas del esquema de desarrollo, se centra en las ventajas *pragmáticas*, en explicar por qué participar y confiar en el software libre tiene sentido desde un punto de vista empresarial. Tanto desarrolladores como ideólogos (Bezroukov, 1999) lo han criticado por omitir o aminorar la motivación ideológica que llevó al que el movimiento naciera y se desarrollara hasta el momento en que el CatB pudo ser escrito; también por limitar su análisis a un conjunto de ventajas y desventajas, incluso provocando un pequeño cisma hacia el interior del movimiento al buscar rebautizarlo con el término *Open Source*, neutro ideológicamente y, por tanto, más aceptable para los empresarios.

El efecto del CatB en el desarrollo del movimiento del software libre es innegable. En 1998 la visibilidad de GNU/Linux comenzó a crecer de manera espectacular, y los dos cuerpos de código más grandes que hoy en día forman parte vital de su éxito en el escritorio[9] fueron liberados partiendo de productos propietarios.

En resumen, las ventajas operativas de este modelo de desarrollo han sido un fuerte motivador para que las empresas se acerquen al desarrollo de software libre, al igual que para muchos desarrolladores como individuos. Y el CatB ha llevado también a un intenso y largo ejercicio comunitario de autoanálisis y definición de posición por parte de cada uno de los participantes a lo largo de un gradiente bien documentado; al grado de que un desarrollador prospectivo puede saber de antemano con cuáles proyectos es más probable que se sienta a gusto colaborando antes de dedicar mucho tiempo a integrarse a su comunidad. Weber (2004, pág. 113) señala:

> Tocó a Eric Raymond [...] escribir el ensayo que formaría el debate central acerca de la identidad. El CatB [...] tuvo un

[9]Los proyectos que forman parte de Mozilla, principalmente representados por Firefox y Thunderbird, liberados por Netscape, y StarOffice, que se convirtió en OpenOffice, liberado por Sun Microsystems tras adquirir StarDivision.

> efecto galvanizador porque articuló una clara lógica en lo técnico y en el comportamiento con la cual la comunidad *Open Source* pudiera identificarse. Los desarrolladores ya hacían todo esto (las prácticas descritas por Raymond); lo que Raymond introdujo fue una justificación convincente acerca de qué hacían y por qué funcionaba tan bien. Produjo también una imagen poderosa y una metáfora evocadora, y lo plasmó de forma concisa, con un lenguaje claro y frases atractivas [...].
>
> El CatB sigue siendo el punto de inicio más socorrido por quien busca entender el fenómeno *Open Source*. El escrito de Raymond es inteligente, a veces brillante y con frecuencia controvertido. [...] No es tan importante si Raymond está o no en lo correcto. Lo que importa es que su trabajo plantó una estaca intelectual para iniciar una discusión explícita acerca de la identidad. Para el movimiento *Open Source*, el CatB marca con claridad el inicio de una nueva fase de madurez, una autoconciencia conjunta, articulada y política alrededor de la cual la comunidad podía (y lo hizo) congregarse y discutir.

Otro punto ampliamente criticado del enfoque del CatB es que la mayoría de los estudios puntuales en que se basa el ejemplo de Raymond, así como los primeros estudios de orientación social que llevaron a cabo líderes del movimiento del software libre entre 1998 y 2002 (Raymond, 1997; Torvalds y Diamond, 2001), se enfocan en estudiar proyectos de amplia fama y repercusión (como Linux, Samba y Apache) o proyectos liderados por personalidades reconocidas y que, a pesar de ser mucho menores en complejidad, tamaño y efecto (por ejemplo, Fetchmail, Git), presentan características diferentes de la mayoría de los miles de proyectos menores que han aparecido. Esto lleva a que la visión general resulte miope, dado que ignora no sólo a gran parte de los proyectos, sino a los desarrolladores. Dice Wuestefeld (2009):

> El bazar de software de Eric Raymond es una fantasía.

> Lo que en realidad ocurre en los proyectos *open source* no tiene nada que ver con su "inmenso bazar burbujeante de diferentes objetivos y acercamientos".
>
> En su clásico *La catedral y el bazar*, Eric nos llama *hordas felices e interconectadas de programadores/anarquistas*. Sí, somos hordas. Pero, ¿somos anarquistas? Más bien, somos trabajadores construyendo pirámides.
>
> ¿Por qué reconocemos nombres como el de Linus Torvalds, Miguel de Icaza y Guido van Rossum? Porque están escritos sobre la entrada de las pirámides de Linux, Gnome y Python, respectivamente. Son los nombres de faraones carismáticos.
>
> Como lo dijo Eric: *Para construir una comunidad de desarrollo, tienes que atraer gente, interesarlos en lo que estás haciendo y mantenerlos contentos con la cantidad de trabajo que deben realizar. El interés técnico te ayudará a lograr esto, pero está lejos de serlo todo. La personalidad que proyectes también importa.*
>
> ¿Y por qué nadie conoce tu nombre? Porque nadie conoce el nombre de un constructor de pirámides.

La dinámica de desarrollo que sigue un proyecto grande y atractivo es necesariamente distinta de la que sigue uno poco conocido y de interés para un reducido grupo de personas. El CatB llevó a mucha gente a la errónea conclusión de que basta hacer público un conjunto inconcluso de fuentes para que alguien se tropiece con él y lo convierta en un proyecto exitoso. Nada más lejano de la realidad; impulsar un proyecto de desarrollo de software libre requiere tanto o más impulso que un desarrollo realizado por completo en casa y con los motivadores tradicionales 100% monetarizados, y esto podemos constatarlo ante la gran cantidad de proyectos abandonados que siguen a la espera de desarrolladores motivados.

En este sentido, me permito relacionar esta disquisición con lo discutido en (Bitzer, Schröder y Schrettl, 2006), acerca de por qué –con base en que el motivador primario del desarrollo de software

es cubrir una necesidad concreta– el desarrollo de software libre recibe un impulso, incluso considerando que, según la teoría de juegos, a todos conviene esperar que *alguien más* haga el trabajo:

> Al permitir que los individuos pospongan sus decisiones por cierto tiempo, por ejemplo, para esperar a que alguien más esté desarrollando el software que ellos requieran bajo un licenciamiento libre, emergen importantes dinámicas. Obviamente, la cantidad de tiempo que un miembro de la comunidad n esté dispuesto a esperar depende del beneficio que recibirá de la existencia del software en cuestión, el costo de desarrollarlo él mismo y sus preferencias de tiempo [...].
>
> Para todo individuo i existe un resultado de equilibrio perfecto del subjuego en el cual únicamente i desarrollará de inmediato. O, más intuitivamente, si nadie más que i desarrolla, entonces la mejor estrategia de i es desarrollar de inmediato, y si i comienza a desarrollar de inmediato, la mejor estrategia de todos los demás es esperar.

De aquí se desprende que, sin restar importancia a este punto, en el desarrollo de software libre, las ventajas originadas puramente por la eficiencia del modelo no son por lo general el único motivador del desarrollo.

4.1.5 La economía del regalo

Para sus conocidos ensayos, Eric Raymond (1997, 2000) partió de describir la sociedad de los *hackers* en términos de su organización respecto a los recursos, como *una sociedad basada en la abundancia*, en contraposición con las *sociedades basadas en la escasez* que acostumbran estudiar las ciencias económicas. A este análisis y con mayor profundidad vale la pena agregar el trabajo de David Zeitlyn (2003).

Explica Raymond que, en sociedades en las que la escasez material no es sensible, el procedimiento para subir en el escalafón

social es dar *regalos* al resto de la sociedad. Esto puede verse en sociedades tribales, en que las familias ofrecen dolorosos sacrificios (por citar el ejemplo clásico, ante la ira de las deidades sacrifican a un miembro de la familia por el bien de toda la comunidad); o en sociedades modernas, que otorgan donaciones públicas y millonarias a organizaciones de caridad o beneficencia.

Un rasgo inherente a las economías clásicas basadas en regalo es que los bienes ofrecidos no pueden juzgarse de modo objetivo o cuantitativo.[10] El regalo es único e irrepetible, creado por su dador y valorado con base en un mérito que puede medirse de muchas maneras, desde su belleza hasta su efectividad.

Como recalca Zeitlyn, en respuesta directa a Raymond, los individuos que forman parte de estas sociedades no actúan meramente por desinterés. *Habiendo dado, habiendo hecho una contribución, la sociedad le debe a dicho individuo.* Esa deuda se constituye en un capital simbólico: Zeitlyn cita en su artículo (2003) a Pierre Bourdieu, quien extendió la noción del capital más allá de lo puramente económico y la llevó a capital cultural y social, capital *simbólico*, que se rige por reglas muy distintas; muy particularmente es capital *no susceptible de intercambio directo.* Cuando mucho, el valor simbólico de un individuo le servirá de impulso al competir ante algún otro recurso en el mercado abierto donde la apreciación subjetiva sea determinante para elegir al beneficiario.[11]

[10] Claro está, en el caso de una beneficencia, la cantidad donada por una persona específica puede ser objeto de comparación. Sin embargo, se tiende a poner más importancia en el valor de la causa hacia la cual se dirige la donación que hacia el valor objetivo crudo de la misma.

[11] Por ejemplo, al sostener una entrevista laboral, en especial ante un evaluador que comprende al funcionamiento de la cultura de software libre, es innegable la ventaja de un colaborador frecuente y prolífico a proyectos de software libre.

4.1.6 Impulso ético-ideológico

No podemos dejar de mencionar la razón principal que dio inicio al movimiento de software libre como tal: la concepción ética-ideológica de cómo *deben* ser las reglas de intercambio en la sociedad al partir de la certeza de que las computadoras forman ya parte fundamental de la sociedad, y que disponer de la libertad de modificar su comportamiento –supuesto básico cuando hablemos de un *bien tangible* de nuestra propiedad– es esencial para el mero funcionamiento de la sociedad.

Los promotores y desarrolladores del software libre por motivos ideológicos coinciden en una percepción cercana al ya conocido discurso relativo a la brecha digital, aunque con un efecto muy distinto. Dado que los países desarrollados cuentan con mayor acceso a la tecnología, y en los modelos hoy prevalecientes de licenciamiento de la llamada *propiedad intelectual* tienen mayores esperanzas de éxito,[12] ven como una necesidad fundamental, inclusive tan básica como un derecho humano más, la creación de software suficiente para permitir a cualquiera la operación de su equipo sin quedar a merced de los designios de una empresa o un reducido conjunto de empresas.

Ese factor, sin duda, llevó a Richard Stallman a crear el proyecto GNU en 1984 y a la Fundación de Software Libre (Free Software Foundation) a lanzarse en 1985 a la reimplementación de una cantidad tremenda de software ya disponible, con la sencilla convicción de que el software en uso no servía al progreso de la humanidad, sino al progreso económico de ciertas empresas específicas, en un área fundamental para nuestro avance colectivo como especie. Y al día de hoy, hacer *lo correcto para todos* todavía es un

[12] Aunque en países con menores índices de desarrollo se impulsen verdaderas campañas de cacería de brujas contra la copia no autorizada, sencillamente no es viable económicamente que la población destine una alta proporción de sus ingresos a pagar esquemas de licenciamiento que pueden calificarse de absurdos y abusivos, por decir lo menos.

poderosísimo motivador del desarrollo de nuevos proyectos y del perfeccionamiento de los ya elaborados.

También ese factor motivó que el líder del desarrollo del sistema operativo OpenBSD,[13] Theo de Raadt, anunciara en mayo de 2001 que, por desacuerdos en el esquema de licenciamiento, OpenBSD no utilizaría ya el filtro de paquetes *ipf*,[14] de Darren Reed (Raadt, 2001); a pesar de ser dicho filtro un componente mayor dentro del principal nicho ocupado por OpenBSD, la comunidad de usuarios y desarrolladores en su mayoría vio esa remoción como algo indispensable. Ante esta necesidad, Daniel Hartmeier, programador suizo que hasta ese momento nunca había desarrollado al nivel del *kernel* del sistema operativo, implementó un filtro de paquetes desde cero en aproximadamente un mes (Andrews y Hartmeier, 2002); esto le valió convertirse en miembro del equipo núcleo de desarrollo de OpenBSD, y 10 años más tarde, *pf* sigue formando parte fundamental de dicho sistema.

4.1.7 Incidencia del proyecto en un área de interés profesional

No podemos pasar por alto uno de los principales factores de motivación, aunque muchas veces sirva sólo como puerto de entrada: una porción cada vez mayor de los participantes en proyectos libres son empleados de alguna compañía, contratados explícitamente para participar en el desarrollo de algún proyecto libre.[15] Y si bien

[13] Sistema operativo libre diseñado para ambientes altamente sensibles a la seguridad.

[14] El subsistema encargado de verificar a cada paquete entrante de la red y actuar conforme a las reglas determinadas por el administrador del sistema, componente fundamental de un firewall implementando un esquema de seguridad perimetral para una red local.

[15] Y esto se explica por la cada vez mayor participación y dependencia de software libre por parte de las compañías. ¡No hay mejor manera de incidir en la dirección de un proyecto que hacerlo desde adentro!

los desarrolladores que ingresan por esta vía no siempre estén ideológicamente convencidos de las virtudes del modelo de desarrollo libre, colaboran en los proyectos en cuestión respetando las reglas del juego y los alimentan con puntos de vista quizá muy distintos, provenientes de otra manera de enfrentarse a sus tareas.

Ya que los tradicionales proyectos de software libre son muy modularizados, muchas veces es sólo cuestión de tiempo para que los desarrolladores se involucren en otros proyectos y se integren al abanico entero del software libre. Y si bien esto hace que, por centro de masa, el enfoque promedio de la comunidad gravite más hacia la parte pragmática que a la ideológica, dicho perfil relativamente nuevo se ha asimilado a las costumbres imperantes en nuestras comunidades.

4.2 Elementos de identificación con la subcultura *hacker*

Un importante elemento que da cohesión a las diversas comunidades de desarrolladores es una gran cantidad de referentes culturales endógenos, que curiosamente han emergido de manera independiente en las diversas comunidades con muy alta concordancia. En muy diversos grupos se observa:

1. Poco reconocimiento a las cualificaciones formales (grado académico, certificaciones otorgadas por la industria, etc.), y una *meritocracia* que si bien no conserva una medición constante por medio del universo de proyectos, se reconoce expresamente como la forma de organización social más adecuada para este medio.

2. Un sentido del humor generalizado, ácido e irreverente. Más allá de las interacciones interpersonales, el humor se hace presente en todos los ámbitos y abarca desde los comentarios

incluidos como parte del código fuente –que en otros entornos estaría enfocado a su compilación o ejecución y no a su uso a modo de medio de expresión–, hasta las publicaciones de corte académico en el medio. Un ejemplo de este punto es la revista *;login:*[16] de la asociación *Usenix*, una de las publicaciones más veteranas en el campo de la administración de sistemas y seguridad en redes, con amplio reconocimiento, que mantiene un tono informal en buena parte de sus artículos y columnas, lo cual, precisamente, se ha configurado en un sello distintivo.

3. Actividades externas al campo con alta dosis de elaboración cognitiva; las reuniones de *hackers* tienden a salpicarse de referencias a literatura fantástica[17] o juegos de alta complejidad cognitiva.[18]

4. Alta politización y relativa homogeneidad entre las posiciones adoptadas por los diversos participantes

Uno de los mayores exponentes de esas características es Larry Wall, principal arquitecto del lenguaje de programación *perl*. En el libro *Programming perl*, los autores (Wall, Christiansen y Orwant, 2000) describen las tres virtudes del programador:

FLOJERA. La cualidad que te hace dedicar grandes esfuerzos para reducir la cantidad total de energía gastada. Hace que escribas programas que ahorren trabajo y que otras personas

[16] http://www.usenix.org/publications/login/

[17] Si bien resulta natural que haya una predilección por la ciencia ficción (bien escrita y fundamentada, acorde con la elegancia y coherencia mencionada), autores que crean mundos fantásticos con suficiente detalle y complejidad, como Tolkien o Pratchett, son ampliamente conocidos y preferidos.

[18] Por ejemplo, el juego de *Mao*, cuyas reglas incluyen no comunicar las reglas; el juego se asemeja a un reto de análisis de protocolo, y una sesión puede durar varias horas, sin mayor incentivo que el de continuar descifrándolo.

encuentren útiles, y documentar lo que escribiste para que no tengas que responder muchas dudas al respecto. Por tanto, ésta es la primera gran virtud del programador, y por esa razón se escribió el presente libro. Véase también *impaciencia* y *vanidad*.

Impaciencia. El enojo que sientes cuando la computadora es perezosa. Esto te lleva a escribir programas que no se limiten a reaccionar a tus necesidades, sino que de hecho se anticipen a ellas. O al menos lo simulen. Por tanto, ésta es la segunda gran virtud del programador. Véase también *flojera* y *vanidad*.

Vanidad. Orgullo excesivo, el tipo de actitud por la cual Zeus te fulmina. También la cualidad que te lleva a escribir (y mantener) programas de los cuales otras personas no quieran hablar mal. Por tanto, ésta es la tercera gran virtud del programador. Véase también *flojera* e *impaciencia*.

Gabriella Coleman escribió en su bitácora personal (Coleman, 2009) una breve reflexión al respecto de este rasgo de la *cultura geek*, motivada por una línea que incluyó en un correo Mark Shuttleworth, dueño de Canonical y líder de la distribución de Linux Ubuntu.[19]

Él inicia un complicado correo acerca de la tensa colaboración entre los proyectos Debian y Ubuntu en un momento clave (Shuttleworth, 2009) diciendo: "Pido disculpas por adelantado si este correo es largo y no particularmente divertido". Coleman comenta:

> En el medio de complejas discusiones técnicas en un congreso o durante una cena, los *hackers* con frecuencia sazonan

[19]Formalmente, y muy acorde con lo que aquí mencionamos, el título de Shuttleworth en Ubuntu es sabdfl: *Self-Appointed Benevolent Dictator For Life*, autoproclamado dictador benevolente vitalicio.

sus conversaciones con una serie de bromas ingeniosas. Si bien bromear es un recurso común utilizado por ponentes en pláticas al público para romper el hielo (al menos en el contexto americano), durante un congreso de *hackers* no sólo son los ponentes quienes bromean; los miembros de la audiencia no dudarán en interrumpir al ponente para introducir un punto de humor, algo que me parece que nunca ofende y es de hecho esperado y celebrado. En otras palabras, el humor prevalece mucho más en su esfera social que en la mayor parte de los otros grupos vocacionales, con la posible excepción de los comediantes.

De tal manera, después de sólo semanas de trabajo de campo, se me hizo aparente, de modo innegable, que el humor es el medio privilegiado a través del cual los hackers expresan su afección cultural por el ingenio y el placer, y se ha vuelto un camino para que yo comprenda la actitud afectiva del placer, la cual de otra manera es tan difícil de capturar analíticamente. El humor, describiéndolo simplemente, es placer, y el juego lo hace socialmente material y tangible. Más aún, dado que el humor *hacker* abarca también con frecuencia asuntos técnicos, funciona como un pegamento cultural que une a los *hackers* en un colectivo social.

4.3 PARALELOS EN OTROS GRUPOS CREADORES

Si bien el presente texto se enfoca a las comunidades de desarrollo de software, repasaremos muy brevemente los posibles paralelos entre ésta y otros grupos sociales que se centran en la actividad creativa y analítica, desde muy distintos ángulos: científicos, artistas y documentadores.

Todos estos grupos están formados por personas insertas en la sociedad general; no por su enfoque dejan de dar importancia y reconocer valor a la retribución económica. Sin embargo, salvo contadas excepciones, quien descubre y persigue una vocación como creador no lo hace por la retribución económica a su trabajo,

sino por recompensas mucho más allá de las expresables por mera lógica de mercado.

4.3.1 Paralelos con la comunidad científica

La comunidad científica presenta grandes paralelismos con las comunidades de desarrollo colaborativo de conocimiento. La principal diferencia es la manera en que un individuo logra ascender para formar parte de esta comunidad. Sin embargo, si bien la comunidad científica cuenta con ritos y procesos de aceptación (estudios y defensa de pregrado, posgrado), una vez que el individuo alcanza el nivel intelectual para ser considerado un par por los académicos, crecen las similitudes con las comunidades colaborativas.

Los académicos empleados por universidades en todo el mundo comparten una característica fundamental con desarrolladores de software libre, autores de artículos de Wikipedia y demás creadores de obras culturales libres: si el valor productivo de un individuo se monetariza, casi invariablemente éste podría conseguir mejor remuneración por su trabajo, al poner acento en el pago por su tiempo, por su propiedad intelectual. Los individuos creadores de conocimiento tienden a valorar muchas formas alternativas de remuneración por encima de la meramente económica. Citando el comentario de Manuel Meza a un borrador del presente texto, en el desarrollo del seminario:

> Para los que se dedican a la ciencia y a producir conocimiento, ése es el fin en sí mismo, no como un medio para llegar a otro fin lejano. La ciencia es reforzante en sí misma. [...] Muchas veces se realiza un trabajo para obtener una remuneración, a pesar de que el trabajo no sea de nuestro agrado pero la paga sí lo sea; lo ideal sería poder trabajar en lo que te gusta hacer y recibir una buena remuneración. En el quehacer científico no es así.

Es cierto que en todo el mundo la producción académica está pasando, desde hace varios decenios, por procesos de evaluación

basados en criterios eminentemente cuantitativos, que derivan en los diversos programas de estímulos a la productividad académica y en premios económicos a los académicos más productivos según determinados criterios. Sin embargo, es de notarse que mucha tinta ha corrido (por parte de los mismos académicos) criticando la arbitrariedad de los referidos criterios, la no homogeneidad de la productividad entre los diferentes campos del saber, la diferencia cualitativa entre obras con el mismo peso evaluable, la demora entre la publicación de un texto y su verdadero punto de mayor efecto, y demás puntos.

4.3.2 Comunidades de creación artística

Las comunidades orientadas a la creación artística son muchas y muy diversas, y probablemente verlas como un gran colectivo sea erróneo desde un principio. Lo que hoy es reconocido como arte, además, enfrenta un severo proceso de cambio y reajuste; las obras artísticas ya no se circunscriben, como hasta hace dos siglos, a una de las siete bellas artes. Por si fuera poco, ni siquiera podemos hablar libres de ambigüedades acerca de lo que significa la *creación*, como bien lo explica Pagola en otro apartado de esta misma obra (Pagola, 2011) y como lo ilustra Spider Robinson (1982). La creación artística, como en las disciplinas científicas e incluso con la libertad que brinda la interpretación, construye sobre lo preexistente. Una transformación o reelaboración hoy es tan aceptada como creación lo mismo que la obra original.

Si bien en su mayoría los artistas[20] de más alto perfil hoy en día quedan excluidos, la proporción en que se presentan dentro del total de sus comunidades es tan reducida que podemos prácticamente tomarlos por ruido estadístico. La mayoría de los artistas[21] acepta esquemas libres o permisivos de creación y difusión de su

[20] Y en particular intérpretes o ejecutores, que distan de ser lo mismo.

[21] Sobre todo cuando acotamos nuestra definición a verdaderos artistas y creadores, no a meros intérpretes, por virtuosos que sean.

obra, incluso sin estar expresamente sensibilizados a esta ideología, por el mero hecho de conocer la dificultad en la difusión de su obra y, por tanto, de su valor en la jerarquía de su comunidad.

Jude Yew, en un reciente ensayo (Yew, 2009), expone numerosos paralelismos entre sus observaciones en la comunidad de creadores de música libre *ccMixter* y los creadores de software libre. En primer término, recalca que un rasgo común entre los participantes de su estudio es el de *no calificar de altruismo* sus contribuciones, sino verlas como un disfrute personal y una inversión:

> A fin de cuentas, toda la actividad que puedes ver en este sitio se traduce en la gratificación personal, o en la esperanza de recibirla, lo cual es el núcleo de toda la expresión. [...] La gente parece comportarse de manera altruista porque espera recibir algo a cambio. Puede ser su propia satisfacción al dar, o puede ser algo proveniente de alguien más. Ver su trabajo remezclado, recibir comentarios sobre su obra a cambio de haber comentado la obra de un tercero, etcétera.
>
> [...]
>
> Yo contribuyo buscando sentir que la gente escucha mi obra.
>
> Hay una cierta satisfacción y validación originadas en las áreas de recomendación y reseña del sitio.

Al igual que entre los grupos de creadores de software, una sensación de pertenencia a la comunidad desempeña también un importante papel (Yew, 2009):

> Lo que me encanta de *ccMixter* es la voluntad de la gente de compartir constructivamente cómo hacen lo que hacen. [...] He aprendido muchísimo a través de las críticas y la disposición a ayudar de los miembros, especialmente a través de las reseñas de lo que he subido.

Retomando esa misma obra, no es de sorprender que los artistas, del mismo modo que los desarrolladores (ya lo hemos explicado aquí), tomen el acto de compartir sus creaciones entre pares como

una conversación, como una forma de expresión que va más allá del lenguaje explícito y verbal:

> Al remezclar el trabajo de alguien estás dando un gran paso. Estás indicándole a esta persona no sólo que escuchaste a su música, sino que tomas su expresión artística y haces algo creativo con ella. Es casi como una conversación [...] Una conversación remezclada donde continúas construyendo y elaborando sobre las ideas de otra persona.

4.4 Diferentes formas y niveles de participación

Muchos proyectos de software libre buscan por todos los medios atraer a usuarios no técnicos a participar en sus filas. Del mismo modo, muchos usuarios muestran una gran motivación para colaborar en diversos proyectos importantes de desarrollo, pero carecen "o sienten carecer" de los conocimientos o del nivel necesarios para hacer contribuciones técnicas. Así, se evidencian variadas necesidades:

1. Es común llamar la atención acerca de que las diversas soluciones basadas en software libre pueden estar muy completas y bien desarrolladas *en inglés*, pero distan de ser aceptables para el grueso de los usuarios si no están disponibles en su propia lengua.

2. Una importante cantidad de programadores se asumen como pésimos documentadores y, además, realizan sus desarrollos por gusto y por interés, sin la disciplina impuesta por un gerente o arquitecto de proyecto con lineamientos respecto a documentación y madurez. Esto hace obvia la necesidad "para el proyecto todo" de que alguien asuma el papel de documentador. Éstos son muchas veces programadores de menor nivel técnico que documentan al tiempo que aprenden

la arquitectura y filosofía general de los sistemas, aumentan sus habilidades y familiaridad con los mismos, y poco a poco aportan código en un círculo virtuoso.

3. Muchos usuarios con inclinaciones artísticas y de diseño se quejan de la apariencia poco profesional o descuidada de diversos programas o componentes, y dedican sus esfuerzos a crear conjuntos de íconos consistentes y fondos de pantalla estéticos, e incluso a proponer, corregir e implementar cambios en la interfaz de usuario.

4. El soporte técnico a los usuarios novatos desde la perspectiva del experto es un papel muy apreciado por los desarrolladores, ya que les permite concentrarse en el desarrollo del proyecto; una comunidad de usuarios entusiastas actúa como una primer red de contención ante las dudas de los usuarios, y su mera participación en ésta suele ser suficiente recompensa para los usuarios avanzados, porque mientras resuelven las dudas de los demás usuarios, ellos mismos aprenden y comprenden mejor el funcionamiento detallado.

5. Diversos proyectos han creado maneras de reconocer el esfuerzo de este perfil de participantes por medio de títulos que revelan compromiso e interés sostenido, como los Embajadores Fedora[22] o los Equipos LoCo Ubuntu.[23]

En las comunidades orientadas al software libre, el eslabón jerárquico más alto lo ocupan los mejores programadores. No necesariamente los más prolíficos, sino quienes tienen mejores capacidades de abstracción, quienes pueden implementar diseños más limpios o *elegantes*, quienes pueden dar cohesión a las necesidades de comunidades muy plásticas y cambiantes de usuarios en todo

[22] https://fedoraproject.org/wiki/Ambassadors
[23] https://wiki.ubuntu.com/LoCoTeamHowto

el mundo: los arquitectos de sistemas en gran escala. Es reconocido, sin embargo (muchas veces difícil de evaluar y distinguir), el importante papel de los colaboradores no técnicos.

Cito ejemplos de proyectos de alto perfil y de personalidades que, desde un papel no técnico, se han integrado fuertemente a los diversos aspectos del desarrollo:

LARRY EWING. Creador del pingüino *Tux* (mascota y logotipo de Linux) y del logotipo de Ximian (empresa dedicada al desarrollo de componentes del escritorio GNOME, adquirida por Novell en 2005).

TY SEMAKA. Ilustrador y músico que desde hace más de 10 años crea las ilustraciones que han dotado de una atractiva identidad al sistema operativo libre OpenBSD. Además, desde 2001 participa con una pieza musical, siguiendo muy diferentes estilos, para cada versión de OpenBSD liberada.

ISMAEL OLEA. Creador y por muchos años coordinador general del Proyecto de Documentación de Linux en Español (originalmente llamado *LuCas*; hoy, *textsctldp-es*).

CHRISTIAN PERRIER. Impulsor del soporte multilíngüe y extraoficialmente coordinador de traducciones del proyecto Debian, y hoy en día desarrollador de Debian; Christian detectó primero la necesidad de contar con soporte multilingüe completo por lo menos en el instalador de Debian, y ha logrado implementar la infraestructura para que hoy en día tengan más de 50 idiomas soportados; en abril de 2007 lanzó el *Smith Project* reconociendo la necesidad de revisar las versiones originales en inglés del software y la documentación elaborada por desarrolladores para quienes el inglés no es lengua nativa.

Sin embargo, no es fácil encontrar la manera de llevar a cabo las contribuciones no técnicas. Con frecuencia una persona que intenta

participar mediante un reporte de fallo, documentación o traducción se siente frustrada al no comprender todos los procedimientos internos, al no ser atendida o solicitársele información difícil de conseguir (Flores, 2009). Atraer la colaboración de los usuarios no técnicos siempre ha sido un reto y una asignatura pendiente para la gran mayoría de los proyectos centrados en el desarrollo.

Además de las comunidades formalmente de desarrollo, debemos mencionar a las comunidades locales de usuarios, incluso sin que participen expresamente en actividades de creación de contenido. Las comunidades de usuarios sirven como primer punto de contacto y como semillero de nuevas generaciones de creadores. Las comunidades locales representan un primer punto de contacto por el cual, casi invariablemente, pasan los futuros desarrolladores al irse familiarizando tanto con la parte técnica como ideológica y organizacional de los proyectos con los que comienzan a contribuir.

Las comunidades locales también exhiben meritocracia y crecimiento, y entre sus miembros suele verse también la necesidad de contribuir con o retribuir a los proyectos que les han brindado respuestas y soluciones a sus requerimientos. La necesidad de aportar frecuentemente se refleja en la pasión con que organizan actividades de difusión en sus comunidades locales, aun sin dar el paso a ser desarrolladores o creadores. Ejemplos de dichas actividades que se realizan periódicamente hoy en día son el Festival Latinoamericano de Instalación de Software Libre (FLISOL)[24] o el Día de la Libertad del Software,[25] con decenas de sedes y miles de participantes en todo el mundo año tras año, así como decenas de congresos, talleres y reuniones de ámbito más local en todo el mundo.

[24] http://www.installfest.net
[25] http://softwarefreedomday.org/SoftwareFreedom.es

4.5 JERARQUIZACIÓN DE LOS INDIVIDUOS EN LA SOCIEDAD

Las comunidades promotoras de la ideología del software libre tradicionalmente se autodefinen como *meritocracias*: grupos sociales en que el principal factor que determina la importancia de un individuo es su mérito, el *valor* (y tal vez la *calidad*) de su contribución. Sin embargo, ¿cómo se mide dicho valor? ¿Cuántas dimensiones tiene el valor dentro de una comunidad? ¿Qué tan relacionado está el crecimiento de una persona en una dimensión y en las otras?

La llamada *comunidad* del software libre dista mucho de ser homogénea, y dentro de ella hay diversos grupos con maneras naturalmente distintas de medir o categorizar los valores. Además, si bien cada uno de estos grupos tiene, con mayor o menor claridad y de manera más o menos explícita, el conjunto de rasgos que determina no sólo la membresía sino los objetivos concretos del proyecto, ni siquiera dentro de cada proyecto se puede predecir directamente el peso específico de un individuo sólo por ver sus contribuciones formales.

4.5.1 SISTEMAS DE REDES SOCIALES

Las comunidades de desarrollo de software libre son verdaderos frutos de internet. Desde los setenta, cuando la explosión del desarrollo universitario de los sistemas Unix derivados de la versión oficial de AT&T, surgieron los grupos de desarrollo sin que mediara más conocimiento interpersonal que el derivado de las listas de correo y del análisis del código compartido. Esto llevó a una verdadera *meritocracia*; algunos nombres de los actores de aquellos años siguen siendo perfectamente reconocibles.

Ahora bien, este desarrollo, si bien relativamente informal, se daba desde el *interior* de las universidades, y el peso como académico de cada persona seguía siendo muy alto. Con la verdadera explosión en el número de personas involucradas gracias a la popu-

larización de internet desde los noventa, y con la diversificación de las actividades que cada uno de los entusiastas realizaba de cara al desarrollo del software libre,[26] el número de actores creció a tal grado que en diversos foros de intercambio[27] se requirió encontrar mecanismos, en un principio, para encontrar el contenido probablemente de mayor calidad. Estos mecanismos al inicio dependían de forma directa de la intervención manual de los participantes e iban orientados a cada comentario. Un punto de quiebre ocurrió cuando a forma de moderación se comenzó a generar *karma*, un mecanismo que prejuzga los comentarios emitidos por cada usuario *basado en su comportamiento previo*, de manera que empezó a crearse una meritocracia mediada por computadora (Malda, 1999), y el sistema conoce y puede indicar a cada usuario su posición relativa en la red.

Advogato[28] nació a finales de 1999 como un experimento orientado a la moderación comunitaria, para demostrar la posibilidad de crear un mecanismo de certificación mutua que no necesitara de intervención humana relativa a cada pedazo de contenido, sino que centrara su operación en cada uno de los usuarios, y capaz de resistir ataques (Levien, 2000; 2009). Partió de una semilla de cuatro entidades altamente confiables: Raph Levien (autor de dicho sistema de métricas) y otros tres *hackers* de amplio reconocimiento, Alan Cox, Miguel de Icaza y Federico Mena.

[26] No es casualidad la baja usabilidad del software (es decir, la facilidad de uso para un usuario no especializado) producido hasta los noventa: Si bien la calidad técnica era del más alto nivel, crear software amigable y apto para usuarios sin amplios conocimientos técnicos no fue prioritario por mucho tiempo.

[27] El caso más emblemático es Slashdot (http://www.slashdot.org), popular sitio de noticias y comentarios relativos no sólo al movimiento de software libre sino a la tecnología en general, que incluye notas no técnicas sobre los intereses culturales prevalentes en esta comunidad y los derechos electrónicos o la ciencia ficción.

[28] http://www.advogato.org/

Mucho antes que fuera práctica común, el sitio *Advogato* ofreció un espacio para que cualquier persona con una certificación mínima (basada en el modelo de Levien) publicara artículos de cualquier temática. Así proporcionaba a cada uno de los usuarios lo que hoy conocemos como un *blog*[29] y, además, *agregaba* o *sindicaba* los *blogs* de todos los usuarios por encima de determinado umbral de confianza.

Si bien el objetivo de *Advogato* era demostrar un modelo de regulación y validación de comunidades en línea, no tardó en convertirse en un referente y en una fotografía del involucramiento de cada persona dentro de la comunidad global de desarrollo de software libre. Al paso de los años, debido a la proliferación de *blogs* personales, que brindan una mayor posibilidad de personalización y no imponen apegarse a una temática ni siquiera por verse como un espacio común, público, *Advogato* ha perdido relevancia y el flujo actual de artículos es ya más bien bajo.

Por su funcionamiento orientado a la certificación mutua, y el comportamiento de muchos de sus miembros de ejecutar determinadas acciones –el envío de un artículo o mensaje, certificar de determinada manera a una persona, solicitar de alguien la certificación–, probablemente hoy calificaríamos a *Advogato* como un sitio que funciona con un esquema de red social (Boyd y Ellison, 2007):

> Definimos los sitios de redes sociales como servicios basados en web que permiten a los individuos: 1) construir un perfil público o semipúblico dentro de un sistema delimitado, 2) articular una lista de otros usuarios con quienes comparten determinada conexión y 3) ver y navegar su propia lista de conexiones

[29] Palabra derivada de weblog, bitácora mantenida en web. Nacieron como diarios personales, pero –gracias al software gestor y al natural efecto de red en internet– se convirtieron en un espacio personal de libertad de expresión para miles de personas en todo el mundo, que permite la agregación en canales de contenidos orientados a la misma temática.

y aquéllas hechas por otros dentro del sistema. La naturaleza y nomenclatura de dichas conexiones puede variar de sitio a sitio.

La importancia de *Advogato*, sin embargo, no consiste en ofrecer un espacio para construir estas redes, sino en *aprovechar su preexistencia* para modelar las relaciones interpersonales ya establecidas, en un espacio mediado por computadora. Con la aparición de los sitios orientados a la creación de redes sociales, muchos desarrolladores de software libre se han involucrado en la creación de varios sitios de esta naturaleza.

Ohloh[30] creado en 2004, merece una mención especial dentro de esos sitios por adoptar un enfoque muy distinto: en vez de invitar a los usuarios prospectivos a llenar sus perfiles mediante las invitaciones de sus contactos, creó los perfiles de los usuarios incluyendo sus calificaciones relativas, con base en el efecto que cada uno de ellos ha tenido en el desarrollo de una gran cantidad de proyectos libres. A finales de 2008 era ya el resultado del análisis de más de 19 000 proyectos (Wikipedia, 2008b). Curiosamente, los actores que *Ohloh* evalúa no lo perciben como un sitio importante o confiable, a pesar de que cuenta con amplias e interesantes estadísticas de actividad e involucramiento de una gran cantidad de personas, y en líneas generales los proyectos a los cuales hace referencia conforman una muestra bastante representativa de la realidad; es decir, no solamente se enfoca a proyectos grandes y exitosos, sino que incluye muchos proyectos pequeños, que representan bien las verdaderas actividades de la mayor parte de los desarrolladores.

Mucha gente ha expresado la falta de confianza en el algoritmo de categorización y peso usado por *Ohloh*. Además, si bien *Ohloh* hace un análisis interesante, al no proveer un verdadero punto de valor a sus usuarios destino (como podría ser un espacio de expresión e intercambio de ideas, en el caso de *Advogato*), no genera

[30] http://www.ohloh.net

la identificación o interés de participar. Por último, no puede dejar de llamar mi atención que *Ohloh* busca atraer a la comunidad de actores del movimiento del software libre, pero está basado en un sistema completamente cerrado. Ofrece un API,[31] pero no documenta públicamente las fuentes y los procedimientos empleados para obtener sus números.

El tema de diseño y análisis de sistemas que representen (incluso, que lleguen a mediar) las relaciones y las jerarquías sociales de esa manera es complejo y merece un amplio estudio; en los últimos años ha habido una gran explosión de sitios web basados en el paradigma de *redes sociales* con muy diversos esquemas para medir y asignar la reputación. Para el que quiera profundizar en el tema, los invito a referirse a Farmer y Glass (2010); incluso a participar en su desarrollo, bajo un planteamiento metodológico no demasiado distinto al del presente Seminario.

4.6 CONCLUSIONES

Los diversos puntos mencionados en este artículo tienden a ocurrir en conjunto. Es muy poco probable encontrar un caso de persona, proyecto o comunidad de desarrolladores que haya crecido y evolucionado mediante uno solo de los factores aquí expuestos; sin embargo, es posible ubicar la participación en un punto del espacio multidimensional hacia el que apuntan estas diversas razones. Comprendiendo la posición relativa de un grupo de individuos, podremos entender la dinámica en un proyecto específico con el que colaboren. Al analizar qué factores llevaron a la creación y cuáles contribuyen a su mantenimiento, o en su defecto, a su estancamiento, entenderemos muchos criterios de diseño que de otra manera no mostrarían una explicación lógica.

[31] Interfaz de aplicación al programador, por sus siglas en inglés; un conjunto de funcionalidades documentadas que una persona externa puede solicitar programáticamente, incorporando la funcionalidad de *Ohloh* en otras páginas.

El reconocimiento de un individuo dentro de su comunidad, sea ésta de la naturaleza que sea, claramente le da un mayor valor como dirigente, y otorga a sus ideas mayor legitimidad y posibilidad de atraer talento para su desarrollo, o interés para su popularización y uso. Ahora, estimar la posición que un individuo ocupa no es algo que en lo que podamos coincidir, salvo en muy contados casos de personas con talento sobresaliente ya sea como desarrolladoras, promotoras o en la esfera en que se les quiera medir. Aunque los sistemas creados a lo largo del tiempo para intentar automatizar esa relación (y que hoy en día mucha gente llamaría sistemas de *redes sociales*) presentan acercamientos interesantes, la única medición confiable y ampliamente aceptada sigue siendo la subjetiva.

Acerca del autor

Gunnar Wolf — México

Ingeniero en software de formación autodidacta; entusiasta, usuario y desarrollador de software libre desde 1997, especializado en la administración de redes y en el desarrollo de sistemas Web.

Ha fomentado la cohesión y profesionalización de las comunidades locales de software libre. Es fundador del Congreso Nacional de Software Libre, y entre 2002 y 2004 se desempeñó como coordinador general del mismo. Además, desde 2003 colabora activamente como desarrollador del proyecto Debian. Es fundador del Encuentro en Línea de Educación, Cultura y Software Libres, mismo que coordinó entre 2005 y 2010.

Desde 2005 trabaja como académico del Instituto de Investigaciones Económicas de la unam.

Bibliografía

Andrews, Jeremy y Daniel Hartmeier (2002), «Interview: Daniel Hartmeier», 2010-05-11, http://kerneltrap.org/node/477; p. 16.

Bezroukov, Nikolai (dic. de 1999), «A Second Look at the Cathedral and Bazaar», *First Monday* 4.12, http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/708/618; p. 10.

Bitzer, Jurgen, Philipp J.H. Schröder y Wolfram Schrettl (mar. de 2006), «Intrinsic Motivation in Open Source Software Development», *Journal of Comparative Economics* 35.1, págs. 160-169, http://www.sciencedirect.com/science/article/pii/S0147596706000643; pp. 5, 12.

Boyd, Danah M. y Nicolle B. Ellison (2007), «Social Network Sites: Definition, History, and Scholarship», *Journal of Computer-Mediated Communication* 13.1, http://jcmc.indiana.edu/vol13/issue1/boyd.ellison.html; p. 30.

Coleman, Gabriella (ago. de 2009), «Sorry, this email is not so clever», 2009/08/06, http://gabriellacoleman.org/blog/?p=1702; p. 19.

— (ene. de 2010), «The Hacker Conference: A Ritual Condensation and Celebration of a Lifeworld», *Anthropological Quarterly*, pág. 26, http://steinhardt.nyu.edu/scmsAdmin/uploads/005/000/UC-proof-Coleman.pdf; p. 7.

Electronic Frontier Foundation (2008), «About the Electronic Frontier Foundation», http://www.eff.org/about; p. 4.

Farmer, F. Randall y Bryce Glass (2010), *Building Web Reputation Systems*, O'Reilly, pág. 336, http://buildingreputation.com/doku.php; p. 32.

Flores, Carolina (dic. de 2009), «A desalambrar el Software Libre», http://piensalibre.net/tics/?p=780; p. 27.

James, Geoffrey (1986), *The Tao of Programming*, Infobooks, http://www.canonical.org/~kragen/tao-of-programming.html; p. 3.

Levien, Raph (2000), «Advogato's Trust Metric», 2009-05-08, http://www.advogato.org/trust-metric.html; p. 29.

— (2009), «Attack-Resistant Trust Metrics», *Computing with Social Trust*, Human-Computer Interaction Series, Springer London, cap. 5, págs. 121-132, http://www.springerlink.com/content/vt330wg160674852/; p. 29.

Malda, Rob (sep. de 1999), «Slashdot Moderation», 2009-05-10, http://slashdot.org/moderation.shtml; p. 29.

Open Source Initiative (1998), «Open Source Definition», http://www.opensource.org/docs/osd; p. 8.

Pagola, Lila (2011), «Esquemas permisivos de licenciamiento en la creación artística», *Seminario Construcción Colaborativa del Conocimiento*, México D.F.: Universidad Nacional Autónoma de México, Instituto de Investigaciones Económicas, http://seminario.edusol.info/; p. 22.

Raadt, Theo de (2001), «OpenBSD CVS: Remove ipf», 2010.2010-05-11, http://marc.info/?l=openbsd-cvs&m=991189189280 72&w=2; p. 16.

Raymond, Eric S. (sep. de 1997), «The Cathedral and the Bazaar», 18/03/2009, http://www.catb.org/~esr/writings/cathedral-bazaar/; pp. 8, 11, 13.

— (2000), «Homesteading the Noosphere», 2009/10/16, http://www.catb.org/~esr/writings/cathedral-bazaar/homesteading/index.html; p. 13.

Robinson, Spider (jun. de 1982), «Melancholy elephants», http://www.spiderrobinson.com/melancholyelephants.html; p. 22.

Shuttleworth, Mark (2009), «On cadence and collaboration», http://lists.debian.org/debian-project/2009/08/msg00092.html; p. 19.

Torvalds, Linus y David Diamond (2001), *Just for Fun: The Story of an Accidental Revolutionary*, HarperCollins, ; pp. 5, 11.

Wall, Larry, Tom Christiansen y Jon Orwant (2000), *Programming Perl*, 3.ª ed., O'Reilly, http://oreilly.com/catalog/9780596000271/; p. 18.

Weber, Steven (2004), *The success of open source*, Harvard University Press, pág. 312, http://www.hup.harvard.edu/catalog/WEBSUC.html; p. 10.

Wikipedia (2008a), *Chaos Computer Club*, http://en.wikipedia.org/w/index.php?title=Chaos_Computer_Club&oldid=249677066 (visitado 04-11-2008); p. 4.

— (2008b), *Ohloh*, http://en.wikipedia.org/w/index.php?title=Ohloh&oldid=209880374 (visitado 10-05-2009); p. 31.

Wuestefeld, Klaus (2009), «The Pyramids and the Bazaar», http://www.advogato.org/article/1020.html; p. 11.

Yew, Jude (2009), «ccMixter: A study of motivations and emergent creative practices that results from open sharing and remixing», *Free Culture Research Workshop*, Harvard Law School, Boston, http://cyber.law.harvard.edu/fcrw/sites/fcrw/images/JudeYewFreeCulture2009Submission.pdf; p. 23.

Zeitlyn, David (jul. de 2003), «Gift economies in the development of open source software: anthropological reflections», *Research Policy* 32.7, pág. 4, http://www.sciencedirect.com/science/article/B6V77-48BC22V-1/2/38734348b0ca9aa0b6d61818b84fbd82; pp. 13-14.