TESIS DOCTORAL

2018

Del *Hau* a la GPL: Una etnografía sobre la constitución de Debian como colectivo sociotécnico.

Fernando González de Requena Redondo

Programa de Doctorado en Diversidad, Subjetividad y Socialización. Estudios en Antropología Social, Historia de la Psicología y de la Educación.

Dirigida por el Dr. Ángel Díaz de Rada Brun.

Esta tesis doctoral ha sido realizada gracias a una Ayuda para la Formación de Personal Investigador (FPI) concedida por el Vicerrectorado de Investigación de la U.N.E.D., en la convocatoria de 2009. Dicha ayuda fue disfrutada desde el curso 2009-2010 al curso 2012-2013, en el Departamento de Antropología Social y Cultural de la U.N.E.D.

Índice general

Índice de cuadros

1 Introducción.

> *I think if we have the right community and the right workflow so that people can work together, if everyone has root, I think that everyone should have root. I think it would be a challenge, but I think it would be possible. If everyone respects each other [...] I don't know how close you can get to that, but I think it is worth trying to get closer than we have now.*
>
> — Clint, Desarrollador de Debian, New York City, julio de 2010.

Esta introducción tiene una doble función. En primer lugar, se pretende tratar algunas cuestiones metodológicas que condicionan la realización del trabajo de campo, en tanto que se ha realizado en buena medida en contextos *online* y mediados computacionalmente. Sin embargo, la conclusión principal de ese tratamiento consistirá en la afirmación de que tal condicionamiento es mucho menor de lo que podría parecer a primera vista.

En segundo lugar, y dada la complejidad del tema y del campo de esta investigación, se trata de ofrecer una visión preliminar y general de este campo. Aparecen así en las dos primeras secciones muchas de las cuestiones en las que se va a profundizar en los restantes capítulos, ofreciendo una idea general del tipo de situaciones que se van a tratar, y una recreación de la experiencia de investigación. En la primera sección se describen tres situaciones que permiten hacerse una idea previa del campo que se va a investigar, de sus características y de alguna de sus dificultades, así como del *ethos* y del *habitus* (Bourdieu, 1977, p. 78 ss.) de

los miembros de Debian. En la segunda sección se describe el acceso al campo en relación a los problemas de investigación que han impulsado este trabajo.

Por último, y tras una sección sobre cuestiones metodológicas, se ofrece un avance de las cuestiones que se tratarán en los siguientes capítulos. Empecemos entonces entrando directamente en la descripción de la acción social que se produce en tres situaciones etnográficas diferentes.

1.1 Una visión preliminar. Tres escenas etnográficas.

Como parte del trabajo de campo en el que se basa esta investigación, en julio de 2009 asistí a la conferencia internacional del Proyecto Debian, DebConf 9, en la ciudad de Cáceres. Esta conferencia se realiza cada año (véase la sección 6.2.1, *DebConfs*), y a ella acuden los miembros del Proyecto Debian que se hallan dispersos por todo el mundo, desde los EE.UU. a Japón, desde Argentina a Finlandia o Nueva Zelanda. Más adelante explicaré en qué consiste y cómo se organiza el Proyecto Debian, pero me gustaría empezar por la descripción de tres escenas que se produjeron en esta conferencia, escenas que nos permitirán introducir una serie de tópicos objeto de la investigación. De momento, basta decir que el Proyecto Debian es una organización que produce una de las distribuciones de *software* libre más usadas y conocidas, Debian. Así pues, se trata de un conjunto de personas que desarrollan y empaquetan *software* libre.

Durante la conferencia, después de la cena (pero en absoluto después del día de trabajo; para muchos de los asistentes, gran parte de la noche es un momento de trabajo), es fácil encontrar cada noche un grupo de asistentes a la DebConf jugando a las cartas en el *hall* de la residencia. Hay de seis a ocho jugadores reunidos en torno a una mesa pequeña y baja. Puesto que los jugadores son de diferentes nacionalidades, el idioma del juego es el inglés. Sin embargo, no se habla demasiado, y los participantes muestran una expresión de concentración y expectación en cada jugada, como si cada lance del juego fuese crucial. Desde fuera, el juego consiste en una serie de rondas en las que cada jugador debe poner alguna sobre la mesa, tomar otras, y decir algo cuando le corresponda. Esta concentración que

observamos no está reñida con el buen humor, y las risas y las caras de sorpresa se producen cada pocas jugadas, en ocasiones tras cada jugada. Cuando esto ocurre, la consecuencia es que algún jugador ha de tomar más cartas del mazo que se encuentra en el centro de la mesa. En ocasiones esto no basta, y tras recoger las cartas, se le indica al jugador afectado que debe tomar todavía alguna más. Mientras algunos jugadores aparentan confianza y seguridad en lo que están haciendo, otros se muestran dubitativos en sus movimientos y sus expresiones.

El juego se llama Mao,[1] y su objetivo consiste en descubrir las reglas del propio juego. Cuando alguien se une al juego, no se le explican las reglas; ha de empezar a jugar, y serán la observación del juego de los demás y las consecuencias de sus propios actos lo que llevará a comprender las reglas que rigen la actividad. Además, según se van sucediendo las partidas, se van introduciendo nuevas reglas, por parte de aquellos que han ido ganando las partidas anteriores. El desafío reside en inventar reglas que se puedan descubrir, para mantener la posibilidad del juego, pero cuyo nivel de dificultad las haga a la vez difíciles e interesantes de descubrir. Es uno de los juegos que es fácil observar entre los miembros de Debian, y uno que nos permite comprender alguna de las concepciones que encontraremos en funcionamiento en sus prácticas sociales. También he encontrado en algunas de las entrevistas realizadas que muchos miembros de Debian usan este juego como analogía de su actividad y sus actitudes.

En primer lugar, podemos descubrir aquí algunas de las características que los desarrolladores de Debian utilizan como signos de autoidentificación: la valoración de la inteligencia y del aprendizaje, en especial el autodidacta, la importancia del reconocimiento por parte de los iguales, la búsqueda de prestigio, el interés por la resolución de problemas, el humor, … En palabras de un jugador experimentado:

> **dato:** El juego tiene un componente cabalístico obvio que se pone de manifiesto cuando uno o más jugadores nuevos se unen a una partida y deben «aprender» las reglas. Los jugadores con experiencia tienen entonces una oportunidad para divertirse en cierta medida a su costa,

[1] Puede verse «Mao (card game)» (Wikipedia, s.f.[a]) o «Mao» (Boardgamegeek.com, s.f.).

> castigándoles cada vez que cometen un error.
>
> Sin embargo, para mí ése es un componente menor (o trivial), y el verdadero meollo del juego se da en partidas en las que sólo participan jugadores que ya conocen el juego: se establece entonces una especie de diálogo o casi batalla intelectual en silencio, en la que uno se prueba intentando deducir lo antes posible las reglas que han introducido los demás.

Es además una situación que pone en juego simultáneamente algunas de las dimensiones que, como iremos viendo, surgen como producto del conjunto de interacciones que se dan en este campo de acción: como tantas cosas en Debian, supone una articulación entre una práctica social (el juego como situación de intercambio social) y una práctica técnica (la inferencia de las reglas y su reconstrucción a partir de su funcionamiento); es al mismo tiempo una práctica instrumental, orientada a fines, en tanto tiene un objetivo claramente definido y que los participantes se esfuerzan en conseguir, y una práctica expresiva, en la que los jugadores expresan su compromiso con los símbolos que les identifican como miembros de la comunidad que antes hemos mencionado.

> **dato:** [...], para un buen juego es imprescindible que los jugadores decidan ejercer un cierto «talante» y actitud relajada. Es increíble lo dado que es Mao a sacar el lado menos amable de las personas ante situaciones de desacuerdo, y éste me parece el aspecto más peligroso del juego. Yo tardé literalmente años en darme cuenta de esto y en poder practicar ese buen talante, y pese a las mucho mejores partidas que disfruté una vez lo conseguí, creo que una parte de las razones por las que ya no juego tiene que ver con tener conciencia constante de ese riesgo, de «si nos descuidamos esto nos explota en la cara».
>
> Pese a todo, considero que el juego es una de las cosas más parecidas que existen a una montaña rusa para el cerebro.

Que un juego, y un juego en el más pleno sentido de la palabra, en el que el dominio convencional de las reglas no es sólo el requisito del juego, sino su obje-

tivo mismo, sea una metáfora adecuada[2] para representarnos algunas particularidades de la actividad que estamos investigando no es seguramente algo casual. Y es que el papel de la diversión es consistentemente invocado como una de las motivaciones para el desarrollo y empaquetado de programas para Debian. El hecho de no limitarse a aceptar las reglas de participación, sino reflexionar sobre, recrear y mejorar constantemente la infraestructura técnico-social de interacción es asimismo una de las señas de identidad de los miembros de Debian. Esta manera de relacionarse con las reglas además los vincula con la figura del *hacker*, como veremos en la sección 4.2.2, *Mediadores e intermediarios. Las figuras del* bug *y del* hacker.

> **dato:** También debe haber un fuerte sentido de la responsabilidad en cada partida: al ser los propios jugadores los que moldean cómo será cada juego, éstos deben ser conscientes de los posibles efectos nocivos (e imprevistos) que sus reglas puedan tener. Muchas veces las partidas pueden acabar hechas unos zorros debido a la *overeagerness* de los jugadores menos curtidos.

Por último, la observación exterior (como fue mi caso) de las partidas de Mao evoca la condición misma del trabajo de campo, incluso más allá de la consideración de la situación social como una forma de actividad regida por reglas: mi pretensión de entender las normas que guían una determinada práctica social no es algo radicalmente diferente del esfuerzo que deben realizar los mismos sujetos cuya práctica intento comprender. Como el jugador novato de Mao, el miembro de Debian se ha embarcado en una forma de vida cuyas reglas son cambiantes, y su primer esfuerzo ha de ser aprender a moverse en este espacio social. Lo que de alguna manera facilita el trabajo del etnógrafo, pues el primer esfuerzo de reflexividad ya ha sido hecho. Más adelante volveremos sobre esta cuestión. Y como el observador externo al juego, el etnógrafo ha de intentar descubrir en qué consiste la experiencia de esta forma de vida (o, por utilizar una metáfora quizás más adecuada con origen en Wittgenstein (2008), de este *juego de lenguaje*).

[2] Para una breve consideración de la analogía del juego en la teoría social contemporánea, véase Geertz (1994, págs. 36-39).

La segunda escena que me gustaría describir se produjo el segundo día de mi estancia en la conferencia. Ese día tendría lugar la primera de tres sesiones de trabajo sobre el *Instalador de Debian*, en las que se reuniría parte del equipo que se ocupa de su desarrollo. Sin entrar en detalles, el *Instalador de Debian* es el programa encargado de instalar el resto de la distribución (del sistema operativo) en el ordenador. Así pues, es una parte crucial en el desarrollo técnico de la distribución. Su desarrollo también es extremadamente complejo. Siendo un encuentro cara a cara entre individuos que suelen colaborar a distancia y a través de sistemas de comunicación y colaboración *online*, pensé que sería una buena oportunidad para estudiar las interacciones sociales en un formato más tradicional. Así que a la hora prevista me presenté en el lugar de la reunión, expliqué mis motivos para estar allí y pedí permiso para asistir a la reunión. Como durante toda la conferencia, nadie puso ninguna objeción a mi observación y me permitieron, como si fuera lo más natural del mundo, observar la reunión y tomar cuantas notas quisiera. Así que, pensando en la suerte que había tenido al escoger como campo de trabajo un grupo tan abierto y que no me había dado más que facilidades a la hora de acercarme a ellos, me dispuse a observar las interacciones que se producían entre ellos al encontrarse cara a cara. Después de algunos meses de seguimiento de las interacciones a través de la red, por fin tenía acceso a un contexto de acción social en el que podía ejercer de etnógrafo de manera más clásica y con algo menos de incertidumbres metodológicas.

Sin embargo, enseguida me indicaron que debía conectarme al canal de IRC (ver cuadro 1.1, *IRC*) que constituía el medio de comunicación habitual del equipo. Y allí estaban mis sujetos, reunidos en la misma habitación tras desplazarse algunos de ellos miles de kilómetros, para seguir comunicándose a través de la red. Con alguien que se encontraba al otro lado de la mesa. Algunos fragmentos ilustrarán la situación (es necesario ver antes los cuadros 1.2, *Cómo leer un log de IRC* y, 1.3, *Cómo leer un log de IRC 2*, para saber como seguir e interpretar la conversación):[3]

```
jul 21 19:22:39 <antonio>  charles: are you here at DC?
```

[3] A diferencia de la comunicación que se produce a través de las listas de correo, la producida en los canales de IRC no se considera pública. Por eso se han cambiado los nombres propios, aunque la siguiente conversación se reproduce con permiso de sus participantes.

Internet Relay Chat (IRC) es una forma de comunicación en tiempo real, basada en texto. Permite comunicarse con multitud de usuarios, bien con un grupo simultáneamente en un determinado canal, bien con un único usuario a través de mensajes privados.
El usuario se conecta mediante un programa llamado *cliente* a un *servidor*, que a su vez está conectado a una red de servidores. Una vez que el usuario se ha conectado a un servidor, puede unirse a uno o más canales para comunicarse con los que se encuentren en el mismo canal. También puede establecer una comunicación privada con los usuarios que se encuentran conectados.
Cada usuario es identificado mediante un *nick*, un seudónimo que el usuario puede registrar y proteger mediante una contraseña, de tal modo que sólo pueda ser usado por él.
Los canales oficiales del Proyecto Debian se encuentran en la red de IRC OFTC («Open and Free Technology Community»), y es posible conectarse a ellos a través de `irc.debian.org` e `irc.oftc.net`.
Para conocer algo más sobre el IRC se puede acudir a «Irchelp.org» (Ircphelp.org, s.f.). Para más información sobre los canales propios del Proyecto Debian, «IRC» (Wiki.debian.org, s.f.[b]).
Las normas de comportamiento esperado en los canales del Proyecto Debian pueden consultarse en «IRC Netiquette» (Wiki.debian.org, s.f.[c]).

Cuadro 1.1: IRC

```
jul 21 19:22:41 <robert>  julian: can you compile the udhcpd binary
in the next busybox upload, please?
jul 21 19:22:54 <julian>  robert: sure
jul 21 19:22:56 <robert>  julian: thx
jul 21 19:22:59 <charles>  antonio: I am. We already talked several
times with each other.
jul 21 19:23:10 <antonio>  charles: so wave :P
jul 21 19:23:11 <charles>  antonio: I'll tell you my nickname the
next time I see you. :)
jul 21 19:23:19 <charles>  antonio: Done. ;-)
jul 21 19:23:30 — [gaston] Remote host closed the connection
jul 21 19:23:30 <antonio>  charles: I'm at hacklab2
jul 21 19:23:30 <charles>  antonio: But you're currently not in my
room where I am. ;-)
jul 21 19:23:31 <antonio>  :P
jul 21 19:23:34 -> gaston (~zxxx@ppp-x-xx.xx-xxx.xxxx.xx) ha entrado
en #debian-boot
```

```
jul 21 19:23:38 <antonio>  charles: crap :P
jul 21 19:23:49 <julian>  robert: dhcp relay support?
jul 21 19:23:59 <charles>  antonio: I'll go showering and then will
show up in the hacklab, too.
jul 21 19:24:06 <charles>  antonio: Just came back from a bike tour.
jul 21 19:24:29 <robert>  julian: not really, just to be sure that
when I will remove the source package udhcp (thus binaries udhcpc
and udhcpd) they will still be provided by busybox binary
jul 21 19:24:36 <antonio>  charles: ok; see you
jul 21 19:24:46 <charles>  Yeah, see you soon.
jul 21 19:24:59 * charles <- gone showering
————————————————————-
jul 21 19:41:57 <antonio>  jon_debconf: around?
jul 21 19:42:01 — [jon_debconf] Ping timeout: 480 seconds
jul 21 19:42:11 <antonio>  i guess not haha
jul 21 19:42:30 <robert>  antonio: he is talking with bob in the
right corner behind you
jul 21 19:42:33 <robert>  bob: ^ ^
```

En realidad, en este primer taller de trabajo del equipo todavía no estaban todos los miembros, algunos de los cuales se encontraban de camino a la conferencia. Entre ellos, el que cumplía el papel de coordinador. La siguiente reunión no recurrió más que a la interacción cara a cara en una reunión presencial, y su dinámica respondió más a lo que yo esperaba este primer día. Pero el extrañamiento, la sensación de falta de familiaridad que es condición de todo trabajo de campo, hacía inevitable la pregunta por la especificidad del espacio social en el que tenían lugar las interacciones sociales: ¿Qué relación se daba entre los contextos *online* y *offline* de interacción? ¿Acaso debía renunciar a seguir a los sujetos más allá del «espacio virtual» y asumir que toda interacción significativa se daba en el marco de la «comunicación computacionalmente mediada»? ¿Estaba el contexto de comunicación cara a cara colonizado sin más por las dinámicas propias de la comunicación virtual (con todas las consecuencias metodológicas que ello implicaría, y que se tratarán más adelante)? ¿Debía aceptar la idea de que las «comunidades virtuales» tienen una dinámica radicalmente diferente de aquellas que se basan en la copresencialidad? A fin de cuentas, ésta fue, en su simplicidad, la primera sensación de *shock cultural*. Y reconocida como tal por uno de los asistentes, un desarrollador con el que había hablado anteriormente. Tras la sesión de trabajo, se acercó a mí y se interesó, preocupado, por si aquello me había parecido demasiado extraño.

A continuación se dan algunas indicaciones para interpretar el fragmento presentado de conversación en un canal de IRC.

- En primer lugar, hay que señalar que lo que se presenta es un fragmento de un *log*, un registro de la conversación. No existen registros públicos de las conversaciones en IRC, cada participante debe guardar aquellos en los que esté interesado. Aquí se presenta un fragmento tal y cómo se guardó en mi ordenador.
- Cada línea corresponde a una intervención. En primer lugar aparece la fecha y la hora en la que se produjo. Luego aparece, entre los signos < y >, el *nick* de la persona que está hablando. A continuación, aparece lo que esta persona dice, visible para todos los que están presentes en el canal. En este fragmento los *nicks* aparecen en negrita. Dependiendo del programa y de la configuración utilizados, es muy posible que aparezcan en diferentes colores, uno para cada *nick*, facilitando la rápida identificación de los interlocutores.
- Cuando alguien quiere dirigirse a alguna otra persona específicamente, es frecuente iniciar la frase con el *nick* de esa otra persona, como en la primera línea de nuestro fragmento, en la que antonio se dirige a charles. Además, normalmente el programa utilizado avisará al usuario, que posiblemente no esté mirando la conversación, de que su *nick* ha sido mencionado.
- Observamos también el uso de *emoticones*, como «:)» (sonrisa), «:P» (sacar la lengua) o «;-)» (guiño), que pretenden complementar una comunicación en principio puramente verbal.
- Vemos también que las conversaciones se van superponiendo. Así, en las primeras líneas se mezclan dos conversaciones, una entre antonio y charles, y otra entre julian y robert.
- Además de lo que cada uno dice, también es visible cuando alguien se conecta o desconecta del canal («`Remote host closed the connection`», «`gaston (~zxxx@ppp-x-xx.xx-xxx.xxxx.xx) ha entrado en #debian-boot`»)
- Si uno de los participantes envía un mensaje precedido por «`/me`», el mensaje aparecerá como «`* Nick mensaje`», como en «`* charles <- gone showering`».

Cuadro 1.2: Cómo leer un log de IRC

- Es posible obtener información sobre cualquiera de los participantes utilizando el comando «`/whois Nick`». Por ejemplo, «`/whois fgrequena`» devuelve:

```
10:04 -!- fgrequena [~fgrequena@4xxxx.telefonica.net]
10:04 -!- ircname : Fernando González de Requena
10:04 -!- channels : #debian-mentors #debian #debconf
#debian-devel
10:04 -!- server : kilo.oftc.net [Zagreb, Croatia]
10:04 -!- away : autoaway after 1500 seconds
10:04 -!- hostname : xx.xx.xx.x
10:04 -!- idle : 1 days 20 hours 40 mins 58 secs
[signon: Sat May 29 13:23:54 2010]
10:04 -!- End of WHOIS
```

- Para mandar un mensaje privado, o abrir un canal privado con otra persona, se pueden utilizar los comandos «`/msg Nick`» y «`/query Nick`».

Cuadro 1.3: Cómo leer un log de IRC 2

Sin embargo, la propia situación sugería algunas respuestas. Lejos de suponer un divorcio radical entre contextos de interacción, el fragmento anterior sugiere la importancia de la continuidad entre ellos. El recurso al IRC no era sólo una manera de situar en el mismo espacio de comunicación a los asistentes a la sesión de trabajo y a quienes no habían podido estar en la misma habitación. Supone además una estrategia de enriquecimiento de la comunicación, en tanto es posible superponer sin confusión diálogos relacionados pero independientes, como se puede comprobar en el fragmento presentado. Además, se articula con interacciones con el espacio *offline* (o como se lo denomina en ocasiones, con el «*meat space*»). Así, podemos ver en el fragmento presentado como se hacen referencias al lugar físico donde se encuentran los participantes: «**`<antonio> charles:`** `are you here at DC? `» (DC significa DebConf, la conferencia en la que nos encontramos). En esta conversación, se contextualiza el lugar físico en el que se encuentran los participantes: «`I'm at hacklab2 `» (un *hacklab* es una habitación donde los asistentes realizan su trabajo, cada uno en su propio ordenador; en la primera Deb-

Conf a la que asistí había dos *hacklabs*), «`he is talking with bob in the right corner behind you`», dado que existe la posibilidad de pasar inmediatamente de la interacción *online* a la interacción *offline*. Esta articulación de espacios *online* y *offline* queda reflejada con claridad en la conversación que hemos visto entre antonio y charles:

```
<antonio> charles: are you here at DC?
[ … ]
<charles> antonio: I am. We already talked several times with each
other.
<antonio> charles: so wave :P
<charles> antonio: I'll tell you my nickname the next time I see
you. :)
```

Además, esta interacción a través del canal de IRC permite su articulación con otros medios de colaboración computacionalmente mediada, como el trabajo común en repositorios de *software* mantenidos mediante diferentes *Sistemas de Control de Versiones*, el trabajo sobre los *bugs* o errores centralizados en el *Bug Tracking System* (o BTS) de Debian, o la contribución paralela al *wiki* del Proyecto (ver Cuadro 1.4, *Algunas definiciones*). De todo ello hay ejemplos en el *log* o registro de la conversación en el canal, aunque por su carácter altamente técnico no se incluyen aquí. Este es un efecto colateral de la comunicación computacionalmente mediada de interés para el etnógrafo: el registro por escrito, de manera automática, de buena parte de las interacciones sociales que se producen.

Por fin, la tercera escena a la que me voy a referir tuvo lugar durante el *Day trip* de la conferencia, un día dedicado al descanso y a alguna excursión por la zona donde se celebra la conferencia. En esta ocasión, el plan era pasar el día en el Valle del Jerte, en la provincia de Cáceres. Tras pasar la mañana en la Garganta de los Infiernos, nos detuvimos unas horas en la piscina natural de Jerte, donde los asistentes se dedicaron a bañarse, tomar algo en el bar que había junto a la piscina, o conversar en pequeños grupos. Y también a llevar a cabo una pequeña *Keysigning Party* o Fiesta de Firmado de Claves.

El significado y desarrollo de una *Keysigning Party* se tratarán extensamen-

Sistema de Control de Versiones: es un sistema que permite la gestión del desarrollo de *software*, guardando copias de las distintas versiones del mismo. Su principal ventaja es que permite la colaboración simultánea por parte de diferentes individuos en una misma pieza de *software*, sin que sus contribuciones entren en conflicto (véase también el cuadro 2.5, *Sistemas de Control de Versiones y Repositorios*).

Bug Tracking System: es una aplicación para registrar y administrar los fallos (*Bugs*) que se producen en un determinado programa, o en un conjunto de ellos. El BTS de Debian permite que comuniquen fallos tanto usuarios como desarrolladores de Debian, y cualquiera puede consultarlo. Se puede acceder al mismo en «Debian bug tracking system» (Debian.org, s.f.[b]) (véase también el cuadro 2.3, Bug Tracking System).

Wiki: Un *wiki* es un sitio web que puede ser editado libremente por aquellos que lo visiten, permitiendo así la existencia de sitios *web* colaborativos. El *wiki* del Proyecto Debian se encuentra en Wiki.debian.org (s.f.[a]).

Cuadro 1.4: Algunas definiciones

te más adelante, en la sección 6.4.2, *Firmado de claves: identidad y confianza*. De momento, basta con señalar que consiste en un proceso mediante el que dos personas se firman mutuamente una clave criptográfica, lo que les permitirá intercambiar documentos cifrados (sobre todo correos electrónicos), además de firmarlos para que no quepa duda de la identidad del que envía el correo, y de otros usos que trataremos más adelante. Como veremos, éste es un proceso fundamental en la creación de confianza en las interacciones *online* en el seno de Debian, y uno de los requisitos indispensables para convertirse en un miembro de Debian de pleno derecho es la posesión de una clave firmada por algún miembro del Proyecto.

En la piscina de Jerte participé en mi primer intercambio de claves, que se va realizando en pequeños grupos. Básicamente, el proceso consiste en asegurarse de que la clave pertenece efectivamente al individuo, y en comprobar algún documento de identidad de la persona, normalmente el pasaporte. Incluso aquí, algunas personas firman directamente la clave en su portátil, que se han traído a la excursión. Esta actividad constituye además un buen motivo para que se comuniquen, siquiera brevemente, algunos de los asistentes que todavía no se conocen.

El hecho de que estén presentes muchas nacionalidades, y haya por lo tanto que intentar comprobar diferentes tipos de documentos de identificación, le da además una dimensión global a la situación. Lo que me gustaría señalar a propósito de esta escena es el contraste entre una actividad fuertemente técnica y el entorno rural en el que se produce, mostrando el hecho de que, para los miembros de Debian, el modo técnico de vida penetra en todos los entornos de acción, conectando de múltiples maneras las dimensiones *online* y *offline*. Por otra parte, me llama la atención el hecho de que el intercambio de documentación para comprobar la identidad de los participantes se produce incluso entre personas que se conocen personalmente y que mantienen relaciones de amistad desde hace años. De nuevo, nos encontramos con una acción social que oscila entre una dimensión *instrumental* (comprobar la identidad burocráticamente documentada) y una dimensión *expresiva* en la que se muestra el compromiso con determinados estándares de seguridad que no son instrumentalmente necesarios. Como veremos, también se ve reflejada aquí la distinción entre los aspectos *técnicos* y los *sociales* de las prácticas en Debian.

1.2 Origen de la investigación y acceso al campo.

Esta investigación surgió de la intención de estudiar las prácticas sociales y simbólicas que constituyen lo que se conoce como la «comunidad del *software* libre». La comunidad del *software* libre está compuesta por individuos separados geográficamente, y que se relacionan en torno a una actividad altamente especializada y tecnológicamente sofisticada, al tiempo que política y éticamente cargada: la creación y distribución de *software* libre. Sobre esta cuestión se irá profundizando a lo largo de todo este trabajo, pero conviene introducir brevemente desde el principio en qué consiste el *software* libre. Desde la antropología, las dos obras más importantes sobre el *software* libre son Kelty (2008) y Coleman (2013). En la primera obra encontramos la siguiente definición del *software* libre:

> Free Software is a set of practices for the distributed collaborative creation of software source code that is then made openly and freely available through a clever, unconventional use of copyright law. But it is

La *Free Software Foundation* (Fsf.org, s.f.[a]) es una organización sin ánimo de lucro creada por Richard Stallman, el fundador del proyecto GNU y el autor de la *GNU General Public License*. También es el autor de numerosos programas de *software* libre ampliamente utilizados.

Su objetivo es «to promote computer user freedom and to defend the rights of all free software users».

Cuadro 1.5: *Free Software Foundation*

> much more: Free Software exemplifies a considerable reorientation of knowledge and power in contemporary society – a reorientation of power with respect to the creation, dissemination, and authorization of knowledge in the era of the Internet (Kelty, 2008, pág. 2).

La principal cuestión que me llevó a interesarme por este campo fue el interés por descubrir y entender las posibles relaciones entre el don y la reciprocidad, tal y como han sido estudiados por la antropología clásica desde Mauss, y la construcción de una forma de relación y organización que podemos llamar «comunidad». Todo ello en un contexto caracterizado por la mediación de las interacciones a través de redes y aplicaciones informáticas, es decir, un contexto en el que los supuestos y las expectativas surgidas de la interacción cara a cara y la copresencialidad no son automáticamente puestos en juego. Este planteamiento supone la consideración del *software* libre como don, como objeto que lleva en sí mismo la expectativa de una forma de reciprocidad que va más allá del cálculo individual y que es capaz de sostener, en su circulación, vínculos interpersonales que desbordan el marco de las interacciones instrumentales. Éste es un tipo especial de *don*: no se da el objeto a las personas que se encuentran en determinada posición social, sino a todo el mundo; no se confía en la reciprocidad para proteger los bienes que se donan de una apropiación, sino en un mecanismo legal; y consiste fundamentalmente en una forma de conocimiento social, en tanto las licencias libres obligan a mantener disponible no el programa informático tal y como se ejecuta en el ordenador, sino el código fuente del mismo.

Como se dice en la página de la *Free Software Foundation* (véase el cuadro 1.5, Free Software Foundation), el *software* libre es «software that gives you the

user the freedom to share, study and modify it. We call this free software because the user is free».[4] En cuanto a la carga ética y política del mismo, podemos leer en la misma página: «To use free software is to make a political and ethical choice asserting the right to learn, and share what we learn with others. Free software has become the foundation of a learning society where we share our knowledge in a way that others can build upon and enjoy». Según la *Free Software Foundation*, el *software* libre se puede definir como aquel *software* que otorga al usuario las siguientes cuatro libertades:

1. La libertad de ejecutar el programa, para cualquier propósito (libertad 0).
2. La libertad de estudiar cómo trabaja el programa, y cambiarlo para que haga lo que usted quiera (libertad 1). El acceso al código fuente es una condición necesaria para ello.
3. La libertad de redistribuir copias para que pueda ayudar al prójimo (libertad 2).
4. La libertad de mejorar el programa y publicar sus mejoras, y versiones modificadas en general, para que se beneficie toda la comunidad (libertad 3). El acceso al código fuente es una condición necesaria.

Como establece la definición, para que estas cuatro libertades sean efectivas es necesario que el usuario pueda acceder al código fuente del programa. El código fuente de un determinado *software* es el conjunto de instrucciones que debe seguir el ordenador durante su ejecución, pero no en el lenguaje que sólo puede entender el ordenador (lenguaje máquina, ilegible para un humano), sino en un lenguaje de programación interpretable también por un humano con los conocimientos necesarios. [5] Incluso, como me decía un Desarrollador de Debian:

> **gwolf:** Me gusta definir el [código] fuente como vehículo de comunicación entre humanos en un lenguaje formal – tan especializado como las partituras, fórmulas u otros.

[4] «What is free software and why is it so important for society?» (Fsf.org, s.f.[b]).
[5] Sobre la definición que utiliza Debian para determinar qué *software* puede ser considerado como *software* libre, véase el apéndice B, *DFSG*.

Una **licencia de *software*** es un documento legal que establece las condiciones de uso y distribución de un programa informático. El autor de un programa informático puede licenciarlo bajo una de las licencias usuales, ya utilizadas por otros programas, o escribir su propia licencia.

Existen dos grandes tipos de licencias, las propietarias y las licencias libres de código abierto (*free and open source licenses*). Las licencias libres son las que garantizan las cuatro libertades mencionadas antes. Además de la GPL, son licencias libres por ejemplo la *BSD license* o la *MIT license*.

Cuadro 1.6: Licencias

El mecanismo utilizado para garantizar las cuatro libertades es el uso de las licencias libres (ver Cuadro 1.6, *Licencias*). No se confía en la obligación moral para mantener los bienes que se donan a salvo de una apropiación privada, sino en un mecanismo legal, la licencia del programa. Si Mauss recurría a la fuerza mística del *hau* para explicar la obligación de no apropiarse definitivamente de lo que se había recibido como don, los creadores de *software* libre prefieren confiar en el dispositivo legal de las licencias libres (la más importante de las cuales es la GPL, *General Public License* (Gnu.org, s.f.[b])). De ahí el título de esta tesis. Lo único que las licencias libres prohíben respecto al *software* al que se aplican es la apropiación privada, originaria o derivada. Es un mecanismo que mantiene así los bienes en circulación. Paradójicamente, son las leyes relativas al *copyright*, diseñadas para garantizar la propiedad intelectual privada, las que aseguran el funcionamiento de este mecanismo. Esto es así porque las leyes de *copyright* conceden al creador del programa la capacidad para determinar las formas de uso y distribución del mismo. Esta determinación se hace mediante la licencia que se aplica al programa, que los usuarios están obligados legalmente a respetar. Pero entre las condiciones especificadas en la licencia puede estar, como ocurre en la GPL, la obligación de no distribuir el programa, o una modificación del mismo, sin conceder a los usuarios las mismas libertades. Sin estos derechos garantizados por las leyes de *copyright*, cualquier persona podría redistribuir un programa libre (o una modificación del mismo) bajo una licencia propietaria. Pero si quiere redistribuirlo, ha de cumplir con las condiciones que el autor ha introducido en su licencia, como establecen las leyes del *copyright*. La GPL tiene por tanto algo de ejercicio de subversión, en tanto utiliza un instrumento (las leyes de *copyright*) diseñado para conceder al autor derechos sobre lo que los usuarios pueden hacer (en defi-

nitiva, para imponer restricciones a éstos), para conceder derechos y libertades a esos mismos usuarios. Y lo hace de tal modo que estos derechos y libertades no pueden ser posteriormente eliminados ni limitados. Se puede entonces sostener la hipótesis de que es el elemento legal (sistémico, societal) el que crea y hace posible el elemento de reciprocidad, de comunidad.

Y se puede hablar de «comunidad» en torno al fenómeno del *software* libre, en tanto existe un grupo social que se autoreconoce como tal:

> What if there were a worldwide group of talented ethical programmers voluntarily committed to the idea of writing and sharing software with each other and with anyone else who agreed to share alike? What if anyone could be a part of and benefit from this community even without being a computer expert or knowing anything about programming? [...] In fact, such a movement exists, and you can be part of it (Fsf.org, s.f.[b]).

No siempre es fácil trazar las fronteras que distinguen las diferentes comunidades que se forman alrededor de este fenómeno, y no es adecuado hablar de «la comunidad» en singular ni en un sólo sentido, pero esto es algo que podemos esperar de cualquier estudio etnográfico que utilice el concepto de «comunidad» en el contexto de las sociedades de la modernidad tardía.

El primer paso consistió en delimitar el campo empírico en el que se iba a desarrollar la investigación. No es posible estudiar directamente la comunidad del *software* libre, en tanto que es un objeto constituido simbólicamente, y de la que no se pueden señalar extensionalmente los sujetos que forman parte de ella. Si el objetivo es la comprensión de los procesos discursivos y simbólicos que la constituyen, no podemos dar por supuesta su existencia como un objeto natural al que podemos acercarnos como observadores. Por lo tanto, debemos buscar algún campo empírico donde estudiar los procesos que la constituyen, esto es, determinar el espacio donde realizar la investigación.

Poco a poco, el espacio social que se fue perfilando para desarrollar la inves-

tigación es el grupo de Desarrolladores de Debian (aproximadamente unos mil voluntarios en todo el mundo).[6] Debian (Debian.org, s.f.[a]) es una de las principales distribuciones de GNU/Linux (ver Cuadro 1.7, *Distribuciones GNU/Linux*), y la que tradicionalmente ha estado más comprometida con la libertad del *software* y con la existencia de una comunidad de *software* libre. De hecho, su documento constitutivo fundamental es el «"Contrato social" con la comunidad de *software* libre».[7]

Otro de los elementos que hacen especialmente interesante a Debian como campo de estudio son sus formas de organización y gobierno, una delicada combinación de democracia y *doocracia*[8] (véase el capítulo 3, *Formas de gobierno. Doocracia, agencia y apertura*). El carácter democrático del Proyecto es algo inusual en los proyectos de *software* libre,[9] y más si tenemos en cuenta su tamaño. Está presente como aspiración, al menos en algunos de sus miembros, la imagen de un proyecto, de una comunidad, donde todo el mundo tiene *root*, como aparece en la cita que abre esta introducción. Tener *root* en una máquina UNIX (un ordenador con un sistema operativo tipo UNIX como Linux; las analogías entre sus características técnicas y el Proyecto Debian son frecuentes, como iremos viendo, especialmente en la sección 4.1, *La Red como metáfora. Unix*) significa tener la capacidad, la posibilidad técnica, de modificar el propio sistema. *Root* es el usuario que tiene todos los privilegios de acceso y modificación en un sistema informático. Que todo el mundo tenga *root* quiere decir que todo el mundo tiene esa capacidad.

Este trabajo de investigación pretende ser pues una etnografía del Proyecto Debian. El Proyecto Debian es la organización que se encarga de desarrollar y publicar el sistema operativo Debian. Los miembros de Debian suelen señalar que Debian es un sistema operativo, un proyecto y una comunidad. De estos tres aspectos y sus relaciones iremos dando cuenta a lo largo de todo este trabajo. Como

[6] El número va variando, aunque desde hace muchos años se ha estabilizado alrededor de esta cifra. Se puede consultar el número exacto en Nm.debian.org (s.f.[b]). Respecto a los colaboradores activos recientemente puede verse Contributors.debian.org (s.f.[a]).

[7] Recogido en el apéndice A, *Contrato social*.

[8] El término se explica en sección 3.2.1, *Los Desarrolladores Individuales: doocracia*. Se refiere al hecho de que aquél que hace algo es el que decide en ese ámbito. Sobre la traducción del término véase la nota 8 del capítulo 3.

[9] Donde suele ser más habitual el papel de alguna compañía o la figura del «dictador benevolente» (véase la sección 3.2.1.1, *Doocracia, meritocracia y democracia*).

Una **distribución GNU/Linux** es un conjunto de programas que se distribuyen juntos de tal modo que sean compatibles y estén bien integrados. Los elementos más importantes son el núcleo Linux (Kernel.org, s.f.) y las utilidades desarrolladas por el proyecto GNU (Gnu.org, s.f.[a]). En definitiva, proporciona un sistema operativo completo de tipo UNIX, junto con una gran cantidad de aplicaciones de *software*.

Cuadro 1.7: Distribuciones GNU/Linux

aparece en su página *web* (Debian.org, s.f.[c]):

> The Debian Project is an association of individuals who have made common cause to create a free operating system. This operating system that we have created is called Debian GNU/Linux, or simply Debian for short.
>
> An operating system is the set of basic programs and utilities that make your computer run. At the core of an operating system is the kernel. The kernel is the most fundamental program on the computer and does all the basic housekeeping and lets you start other programs.
>
> Debian systems currently use the Linux kernel. Linux is a piece of software started by Linus Torvalds and supported by thousands of programmers worldwide.
>
> However, work is in progress to provide Debian for other kernels, primarily for the Hurd. The Hurd is a collection of servers that run on top of a microkernel (such as Mach) to implement different features. The Hurd is free software produced by the GNU project.
>
> A large part of the basic tools that fill out the operating system come from the GNU project; hence the names: GNU/Linux and GNU/Hurd. These tools are also free.
>
> Of course, the thing that people want is application software: programs to help them get what they want to do done, from editing documents to running a business to playing games to writing more software. Debian comes with over 25000 packages (precompiled software that is bundled up in a nice format for easy installation on your machine) — all of it free.

> It's a bit like a tower. At the base is the kernel. On top of that are all the basic tools. Next is all the software that you run on the computer. At the top of the tower is Debian — carefully organizing and fitting everything so it all works together.

Una vez decidido el campo empírico en el que iba a desarrollar la investigación, había que acceder al mismo. La primera dificultad consistía en la inexistencia de un lugar físico al que aproximarse. Se imponía, pues, el recurso a la etnografía virtual como modo de acceso al campo. Veremos en la sección 1.3, *Cuestiones de metodología*, algunos de los problemas específicos que plantea esta aproximación al campo. De momento, se puede señalar que, respecto a los grupos que se relacionan a través de redes informáticas públicas, estamos en una situación en la que podríamos perfectamente cumplir el viejo sueño naturalista de observar sin influir sobre aquello que observamos. Se pueden observar encubiertamente las interacciones sociales *online* sin alterarlas, porque siempre existen ya lo que se llama *lurkers* (merodeadores), sujetos que observan las interacciones *online* sin participar. Cualquiera de ellos podría perfectamente ser un observador realizando etnografía encubierta y no declararlo; nadie se extrañaría. Esta forma de actuación tiene implicaciones respecto a la ética de la investigación social, como señala por ejemplo Hine:

> Aquí, si el investigador asume que las interacciones *online* son suficientemente reales como para dar sustento a un contexto de estudio etnográfico, y es coherente, tiene que aceptar también que los participantes puedan sentirse agredidos, engañados o invadidos en su privacidad, por él o por su investigación. He ahí una consecuencia ética (Hine, 2004, pág. 36).

Es decir, no renunciar a las ventajas del anonimato implicaría en este caso renunciar a la formación de la sensibilidad etnográfica. Desde un punto de vista metodológico, nuestro problema es el contrario, construir una presencia en el campo como etnógrafos que sea epistemológicamente productiva.

Puesto que Debian es un proyecto basado en el trabajo voluntario, desde el principio pareció claro que la mejor manera de construir una presencia en el campo que me permitiera ejercer la observación-participante era encontrar una manera de colaborar en el Proyecto. Y dado que en el origen del proyecto de investigación estaba el interés por descubrir las relaciones entre el don y la construcción de la comunidad, buscar una entrada al campo de esta manera no es sólo una necesidad práctica, sino una fuente imprescindible de conocimiento etnográfico. Buscar una forma de colaboración no permite solamente entrar en el campo, sino que iluminará lo que ha llegado a ser una de las preguntas principales de esta investigación: ¿qué significa ser un miembro de la comunidad Debian, y cómo se llega a serlo? El camino de acceso al trabajo de campo puede empezar recorriendo los puntos de acceso a Debian por parte del usuario interesado.

Y el primer paso es la página principal del Proyecto Debian (Debian.org, s.f.[a]). Lo primero que llama la atención al acceder a la página es que, aun sin solicitarlo, ésta aparece en castellano. Es decir, en la jerga informática, aparece una página *localizada*. Los procesos de *internacionalización* y *localización* (usualmente abreviados como «i18n» y «L10n»; los números 18 y 10 aluden al número de letras de los términos *internationalization* y *localization*) son el medio para adaptar el *software* a diferencias lingüísticas y culturales. Este proceso de localización va más allá de una mera traducción, en tanto

> include[s] adaptation of graphics, adoption of local currencies, use of proper forms for dates, addresses and phone numbers, the choices of colors and many other details, including rethinking the physical structure of a product. All these changes aim to recognize local sensitivities, avoid conflict with local culture and habits (Wikipedia, s.f.[b]).

La combinación de estos dos procesos es denominada en ocasiones *Globalización*.

Ya en la página principal del Proyecto Debian, aparece un acceso directo a la página «Cómo puede ayudar a Debian?» (Debian.org, s.f.[d]). Lo primero que aparece en esta página, y que mostrará su relevancia para entender el sentido del

término *comunidad* en Debian y la concepción acerca de lo que es un miembro de Debian, es la distinción entre usuarios «expertos» e «inexpertos». Ambos «pueden ayudar». Tras listar los campos en los que es posible colaborar con Debian aparece lo siguiente:

> Hay, como puedes ver, muchas formas en las que puedes involucrarte en el proyecto y sólo para algunas de estas es necesario ser un desarrollador de Debian. Muchos de los proyectos existentes tienen mecanismos que permiten el acceso directo a los árboles de código fuente a los colaboradores que han demostrado que son valiosos y de confianza. Generalmente aquellas personas que piensan que pueden involucrarse mucho más con Debian se unirán al proyecto, pero ésto no es realmente necesario.

Aquí se deja claro que colaborar y participar en el Proyecto no requiere ser Desarrollador, miembro de pleno derecho. Ser Desarrollador supone, básicamente, el derecho a votar, el acceso a la infraestructura (las máquinas) de Debian, una dirección de correo «*nombre*@debian.org», la capacidad de subir paquetes de *software* al archivo de Debian (aunque hay excepciones),[10] y la subscripción a la lista de correo privada de los desarrolladores de Debian. Aquéllos que quieran colaborar en la actividad básica de los Desarrolladores, el mantenimiento de paquetes de *software*, deberán poseer «mucha experiencia», además de haber «demostrado que son valiosos y de confianza». En el párrafo citado, la frase «se unirán al proyecto» es un enlace (Debian.org, s.f.[e]), que se llama «cómo puede ayudar» (en castellano; la versión en inglés, se llama «How You Can Join» (Debian.org, s.f.[f])), título muy similar a la que acabamos de considerar, «¿Cómo puede ayudar a Debian»? Pero esta página se dirige a los futuros Desarrolladores desde el principio. De momento, basta señalar que la posible colaboración se puede realizar, y se espera que se realice, de múltiples maneras, con diferentes requisitos, privilegios y exigencias. Pero lo que es más importante, que la colaboración, y por lo tanto la participación en el Proyecto, son siempre anteriores a la pertenencia al mismo, que no es necesario ningún trámite previo a la colaboración misma.

[10] Sobre todas estas cuestiones y los cambios que se han producido a lo largo de la historia de Debian, véase el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*

Este es precisamente uno de los rasgos más llamativos del Proyecto Debian: la apertura del mismo. Apertura en dos sentidos, ambos importantes para el etnógrafo que se acerca a investigar. En primer lugar, la apertura de la gran mayoría de sus canales de comunicación y, por consiguiente, de las prácticas sociales y de los procesos de decisión que nos interesan como etnógrafos. Los canales de comunicación más importantes son las listas de correo, y determinados canales de IRC. Si la mayoría de éstos son públicos, en el caso de las listas de correo los mensajes, además, se archivan y se consideran públicos, lo que permite el acceso a los mismos a cualquiera mucho tiempo después. En palabras de Martin Krafft, un Desarrollador de Debian:

> As part of its commitment to its users, Debian makes operating system development completely transparent to the public. Discussions related to design choices in packaging and other issues are held publicly on the debian-devel mailing list, and contributions are not restricted to developers. In fact, it is quite common for interested users to join these discussions and contribute their thoughts and suggestions. It cannot be stressed enough that Debian would be nowhere if it were not for the massive input the project has received from its user base. Rather than developing for their users, the Debian developers lay open their cards and work on the Debian system together with their users. Of course, participation in the project is not required to use the Debian system (Krafft, 2005, pág. 38).

Pero sobre todo, apertura en el sentido de que cualquiera puede empezar a colaborar en el Proyecto sin ser invitado, y sin que nadie tenga que aprobar la propia actividad. No se entra en el Proyecto para empezar a colaborar, sino que es la propia actividad ya realizada la que va abriendo puertas y proporcionando privilegios al colaborador.

El hecho de que el Proyecto esté realmente abierto a la colaboración está estrechamente ligado al sentimiento, expresado por muchos Desarrolladores de Debian, de que es necesario devolver algo a la comunidad que desarrolla el *software* que se utiliza (véase la sección 5.1.2, *¿Por qué se dona el* software *libre?*).

Esta necesidad, expresada consistentemente en las entrevistas realizadas, se ve reforzada por el hecho de que el canal de vuelta, desde el usuario al desarrollo de la distribución, está efectivamente abierto (véase la sección 5.3, *La obligación de recibir*). Veamos dos ejemplos de las entrevistas realizadas a dos Desarrolladores:

> **ana:** Es que vamos a ver, en Debian pasa una cosa, [...] mientras alguien quiera trabajar en el *port* de Debian para que sea ..., [para que] en lugar de con Linux funcione con Hurd, tú tienes los recursos del sistema igual que cualquier otra persona. [...] ¿Qué es lo que pasa? Que las distros comunitarias, por ejemplo Debian o Gentoo, mientras haya gente con ganas de trabajar algo, tienes los recursos para buscar otra persona.

> **dato:** Llegado un punto supe que yo quería colaborar con esto, yo quería formar parte de eso, y entonces cuando conseguí tener internet en casa, [...] empecé a dedicarme a colarme en la comunidad. Me apunté a muchas listas de correo y empecé a leerlas, a leer como funcionaba todo eso. [...] Yo siempre digo que son cuatro pasos para formar parte de una comunidad de *software* libre, lo digo en inglés, *lurk*, *learn*, *do*, and *become*. Primero *lurk*, que es como estar ahí, escondido, leyendo mucho las discusiones entre la gente, para ver cómo se hacen las cosas, para ver cómo funcionan; el siguiente paso es *learn*, aprender la tecnología que se necesita; luego *do*, hacer, hacer muchas cosas, empezar a colaborar, a proporcionar cosas útiles; y entonces el último paso es *become*, convertirse en miembro de la comunidad. [...] Hice mucho *lurking*. [...] Cuando ya me sentí cómodo [...], intenté empezar el paso hacer, el *do*. [...] Elegí dos paquetes que yo usaba, e intenté ayudar con ellos, que eran `mutt`, y `vim`. Empecé en el BTS, intenté hacer como *cleaning*, ver cuáles eran pertinentes, ver cuáles ya se habían arreglado y cerrarlos, contactar a los que habían informado del error para ver si todavía lo sufrían, entonces es curioso, recibí una pequeña nota del encargado de `vim` diciendo que prefería que no lo ayudara, que prefería él estar a cargo [...] Pero el mantenedor de `mutt` no me dijo nada. [...] Entonces fui haciendo cosas en el BTS [...].

Así que, en enero de 2009, empecé a colaborar en la traducción de algunas partes de Debian, en concreto las plantillas utilizadas para traducir los mensajes que el sistema realiza al usuario para configurar un programa en el momento de su instalación (plantillas `po-debconf`). Como es usual dentro del Proyecto Debian, la coordinación de la traducción se hace a través de una lista de correo, en este caso `debian-l10n-spanish@lists.debian.org` (Debian-l10n-spanish, s.f.). Es decir, la lista de localización de Debian en español. Puesto que esa colaboración tenía el interés de permitirme entrar en el campo para realizar una investigación, lo primero que hice fue presentarme en la lista explicando mis intenciones. La respuesta fue darme la bienvenida (y algunos consejos sobre como empezar la traducción) como colaborador, pero también como investigador. Algunas personas incluso muestran su interés y su disposición a colaborar en la investigación. El Proyecto aparece al investigador como un campo abierto a su investigación.

Pero lo que es más importante, aparece como un campo abierto a aquél que quiere colaborar en el desarrollo del Proyecto. Ser parte del equipo de traducción no requiere más que suscribirse a la lista de correo y hacer las traducciones.[11] La mecánica del trabajo consiste en buscar un elemento que al traductor le interese traducir, comprobando que no esté ya traducida. Para ello, se consulta en las páginas *web* del Proyecto que indican el estado de traducción de cada elemento. Una vez elegido, se manda un mensaje a la lista indicando la intención de traducirlo, para evitar la duplicación del trabajo. Una vez hecha, la traducción se manda a la lista pidiendo revisiones y correcciones, y los participantes contestan en un plazo de siete días (más otro de tres cuando se considera que la traducción es correcta). Se suelen ofrecer correcciones ortográficas y de puntuación, de términos, a veces hay una pequeña discusión sobre cuál es la traducción más adecuada de alguna expresión, pero lo que me interesa destacar aquí es que es el traductor que ha iniciado el proceso el que al final decide cuándo y cómo la traducción es correcta, y el que se encarga de enviarla al mantenedor del paquete que está traduciendo, a través del *Bug Tracking System*. El traductor es en definitiva el responsable de su trabajo. Además, su nombre y su dirección de correo quedan incorporados a la traducción, de tal manera que cuando sea necesaria una corrección o actualización de la misma pueda ser fácilmente localizable.

[11] Sobre la apertura y la facilidad a la hora de empezar a traducir, pueden verse por ejemplo los mensajes Debian-l10n-spanish (2010a) y Debian-l10n-spanish (2010b).

De alguna manera, esto le da una suerte de «propiedad» sobre la traducción, de tal modo que a partir de ahora será el encargado de mantener la traducción. Veamos un ejemplo que aparece en un mensaje a la lista de correo de traducción al castellano:

> Me hago cargo
>
> [...] agradezco tu empuje e interés, no quiero que te tomes esto mal pero sí que consideres los siguientes puntos como una crítica constructiva:
>
> - cuando se envíe una solicitud de actualización de una traducción ya existente (y que alguien «vivo» mantiene) deberías dejar que el traductor anterior la realizara. Al fin y al cabo, es su responsabilidad hacerlo. Sólo debe intervenir un nuevo traductor cuando la traducción esté muy desactualizada (y lleve mucho tiempo así) o cuando la traducción sea muy deficiente.
> - Me sorprende, también, que añadas tu nombre al (C) de la traducción, no creo que los cambios justifiquen esto. Si te fijas el autor original de la traducción no puso una marca de (c) propia sino que cedió el (c) a SPI. Igualmente, has modificado la cabecera Last-Translator que indica al traductor que la mantiene. De alguna forma «apropiándote» de una traducción que otro debía mantener. Eso no deberías haberlo hecho.
> - Para terminar y aclarar otro tema: en general, las actualizaciones de traducciones no hace falta que pasen (salvo que se hagan muchos cambios en el contenido original) por el proceso de revisión porque se supone que los cambios son menores.
>
> Es posible que hayas hecho algunas de estas cosas sin pensar que podía suponer un problema. Sólo quiero hacer ver que la forma de trabajar con traducciones ya realizadas/asignadas no es el mismo que con traducciones que nadie ha realizado aún y que están «libres».

En definitiva, lo que quiero indicar con todo esto es que, por un lado, colaborar en el Proyecto no tiene más condiciones que empezar a hacerlo. Esto es válido tanto para quién realiza traducciones sin ser miembro del Proyecto, como para el Desarrollador experimentado que quiere introducir alguna innovación o trabajar en determinado ámbito del Proyecto. En la mayor parte de las ocasiones, no tiene que pedir permiso a nadie, del mismo modo que nadie le puede imponer la realización de una determinada tarea. Por otro lado, cada individuo es el responsable de sus aportaciones al Proyecto, y de alguna manera el «propietario» *de facto* de su parcela en el mismo. Ambas ideas quedan recogidas en el concepto de «doocracia» (véase la sección 3.2, *Toma de decisiones: de la meritocracia a la doocracia*), un término que cada vez es más utilizado por los miembros de Debian para referirse a una de las formas de gobierno del Proyecto (junto con la democracia y la meritocracia). En palabras del *Debian Project Leader* elegido en abril de 2010: «we operate as a do-ocracy: anyone willing to do things can decide what and how they are done, and no one can be forced to do anything» (Debian.org, s.f.[g]).

1.3 Cuestiones de metodología.

Hemos visto en la sección anterior que se trata de investigar la vida social de individuos dispersos geográficamente y cuya interacción se produce en gran medida gracias a diversas mediaciones tecnológicas. En qué sentido se puede hablar en estas condiciones de comunidades es una discusión que dejamos para el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?* Pero esto nos lleva ahora a la necesidad de plantearnos qué implicaciones y consecuencias metodológicas tiene esta situación.

El trabajo de campo de esta investigación se basa en dos grandes ámbitos de observación y registro etnográficos. Por un lado, la observación presencial de la acción social en reuniones y conferencias de los miembros de Debian, las *DebConfs* (Conferencia internacional de Debian; véase la sección 6.2.1, *DebConfs*), de los años 2009, 2010 y 2011 en Cáceres, New York y Banja-Luka, y en la DudesConf (reunión más informal de miembros y colaboradores españoles, aunque también con presencia de miembros de otros países) de 2010, en La Coruña. En estas conferencias, aunque no sólo allí, se realizaron también casi todas las en-

trevistas presenciales en profundidad a Desarrolladores de Debian que aparecen citadas a lo largo de todo este trabajo. El nombre con el que aparecen identificados estos entrevistados es el nombre que suelen usar en sus comunicaciones en el Proyecto y por el que son conocidos. Suele ser el *nick* usado en IRC o el nombre que aparece en las direcciones de correo.

Por otro lado, la observación y registro de la acción social realizada a través de los canales de comunicación públicos del Proyecto Debian. Éstos incluyen las listas de correo (véase cuadro 2.1, *Listas de Correo*), canales de IRC (véase cuadro 1.1, *IRC*), el BTS (véase cuadro 2.3, *Bug Tracking System*), diferentes *blogs* y agregadores de *blogs* (Planet.debian.org, s.f.), la página *web* del proyecto (Debian.org, s.f.[a]) y su *wiki* (Wiki.debian.org, s.f.[a]), la documentación producida por el Proyecto y sus miembros, etc. Todas estas fuentes son enormemente ricas y se han usado y citado frecuentemente en este trabajo. La excepción aquí es el registro de las conversaciones mantenidas en los canales de IRC. Aunque han sido muy útiles en la presente investigación, dado su estatus no propiamente público[12] no se han incluido citas de esos canales, a excepción de breves fragmentos como los incluidos en esta introducción, sobre los que se pidió permiso explícito y se cambiaron los nombres. Otra excepción son los *logs* registrados con *MeetBot* y hechos públicos (véase la sección 2.1.5.1, *El robot* Meetbot).

Las entrevistas se circunscriben por tanto a un periodo de tres años, y se refieren a la situación en esos años. En algunas secciones se citan extensamente estas entrevistas (así como correos electrónicos o *blog posts*). Esto es así porque la reflexividad y el interés de los entrevistados por sus prácticas sociales arrojan mucha luz (también en la forma de expresarse) sobre la acción social estudiada y sus implicaciones, y permite ir muchas veces más allá de lo explicado en la sección correspondiente. Los documentos analizados y citados corresponden a un periodo de tiempo más amplio, pues se han podido seguir consultando según se redactaba este trabajo. Cuando el marco temporal sea relevante se hará constar, teniendo en cuenta por ejemplo que cuando se indica que alguien ocupa un cargo se refiere al momento de la entrevista o de los documentos analizados. Hay acontecimientos

[12] Véanse las indicaciones sobre el uso de los canales de IRC en Wiki.debian.org (s.f.[c]), donde se establece explícitamente «No public logging». Para una discusión sobre la deseabilidad o inconveniencia de registros públicos de los canales de IRC, véase la discusión en lista de correo que sigue a Debian-vote (2017).

muy relevantes para la historia del Proyecto Debian en estos últimos años que no se describen en detalle, porque supondría una extensión mucho mayor, aunque se alude a alguno y se proporcionan referencias sobre los mismos.

Así, por ejemplo, después de acabar el trabajo de campo intensivo surgió una controversia en Debian que enlaza muchos de los problemas que se van a tratar aquí: la naturaleza del *software*, la distinción entre lo social y lo técnico, los procesos de decisión, la doocracia, las votaciones, el funcionamiento del comité técnico, el consenso, las relaciones con otras distribuciones y sobre todo con Ubuntu, las necesidades de los usuarios, la posibilidad de elegir, y las interrelaciones de todo ello. Fue una controversia sobre un tema aparentemente poco problemático para el profano en estas cuestiones: cuál era el sistema de inicio del ordenador, el primer proceso que se ejecuta cuando el *kernel* se ha cargado. Aunque de hecho se han introducido en esta investigación temas y referencias posteriores, el tamaño de esta controversia y sus implicaciones ha hecho imposible analizarla y considerarla adecuadamente. Para el lector interesado, dos referencias para empezar a entender la cuestión podrían ser Corbet (2013) y el informe de *bug* correspondiente, «tech-ctte: Decide which init system to default to in Debian» (Bugs.debian.org, s.f.[b]), además por supuesto de las discusiones en las listas de correos del Proyecto.

Significativamente, la importancia relativa de los encuentros cara a cara y de la observación y comunicación computacionalmente mediadas para el etnógrafo, al menos en este caso, refleja la experiencia de algunos miembros de Debian, como veremos en la sección 6.2, *Interacción entre aspectos* online *y* offline. *DebConfs*. Allí veremos que las primeras interacciones suelen ser por medios digitales, y cómo se transforman después de asistir a alguna conferencia o encuentro presencial. En mi caso, cuantitativamente ha sido muy importante la observación *online*, pero ésta ha sido mucho más comprensible a partir de los encuentros y la observación presenciales.

Veamos ahora algunas consideraciones sobre la observación y registro de la acción social producida a través de estos dispositivos.

1.3.1 Sobre el registro de la acción social mediada computacionalmente.

Lo primero que hay que señalar es la enorme cantidad de material a disposición del etnógrafo: archivos de decenas de listas de correo (se han seguido regularmente unas cuantas de ellas, y esporádicamente muchas otras), con hilos de discusión que alcanzan en ocasiones decenas de mensajes; registros privados (y algunos públicos) de canales IRC donde hay conversaciones continuas; documentación técnica y no técnica del proyecto; *blogs* de miembros y otros; o vídeos de una multitud de charlas y encuentros en conferencias del Proyecto, por citar sólo algunas fuentes de material documental. En cualquier caso, un volumen que hace imposible un análisis exhaustivo de los mismos.

Todo este material, gracias a su abundancia pero también a su interrelación y, como veremos, a la riqueza de los marcadores contextuales que quedan también archivados, transforma la observación etnográfica de la acción social. Kelty lo indica también con claridad:

> Indeed, as a methodological aside, one reason it is so easy to track such stories and narratives is because geeks like to tell and, more important, like to archive such stories –to create web pages, definitions, encyclopedia entries, dictionaries, and mini-histories and to save every scrap of correspondence, every fight, and every resolution related to their activities. This "archival hubris" yields a very peculiar and specific kind of fieldsite: one in which a kind of "as-it-happens" ethnographic observation is possible not only through "being there" in the moment but also by being there in the massive, proliferating archives of moments past (Kelty, 2008, págs. 114-115).

El campo del presente trabajo de investigación se constituye así en buena parte a través de textos (de todo tipo) objetivados en redes de comunicación y colaboración *online*. Además, el hecho de que los canales de comunicación del Proyecto Debian sean, en su mayoría, públicos y abiertos, implica que la investigación etnográfica sobre Debian, y en gran medida sobre el mundo del *software*

libre en general, se ve transformada en gran medida, y que la construcción del texto etnográfico depende mucho más de la colaboración implícita (dada la apertura general del Proyecto Debian, como hemos visto) o explícita (esa apertura es puesta en ocasiones, por parte de los propios miembros de Debian, en relación explícita con la posibilidad de la investigación social sobre Debian). Los materiales textuales están ya disponibles previamente, y seguirán estándolo públicamente tras la reunión y ensamblaje de estos textos en el texto etnográfico. De ahí la importancia y la abundancia de los enlaces que aparecen.

Como señalan Marcus y Fischer,

> Rather we step into a stream of already existing representations produced by journalists, prior anthropologists, historians, creative writers, and of course the subjects of study themselves. And, therefore, a primary framing task of any ethnography is to juxtapose these preexisting representations, attempting to understand their diverse conditions of production, and to incorporate the resulting analysis fully into the strategies which define any contemporary fieldwork project. In a sense, it is this need to incorporate the field of representations as existing social facts into the anthropologists' practice of ethnography that impels both a multisited terrain for the latter and new norms and recognitions for the relationships so central to the tradition of fieldwork (Marcus y Fischer, 1986, pág. xx).[13]

Por supuesto, esto es algo propio de toda etnografía. Pero en este caso, esas representaciones, esos textos no son el producto de un proceso previo, parte del trabajo etnográfico, de textualización de prácticas e interacciones sociales: son además, ya en su producción como textos, prácticas e interacciones sociales en sí mismos. Y es que al acceder a los *logs* de IRC o a los mensajes archivados de las listas de correo, estamos accediendo a las interacciones sociales tal y como aparecen ante quienes las producen. Leer una lista de correo no es simplemente leer un archivo que objetiva una acción social previa: es observar (parte de) la

[13] Sobre esta cuestión, y sus implicaciones para pensar la coautoría entre etnógrafo e informante, véase también Strathern (2004, págs. 11-13) y Tyler (1986).

acción social misma. Los textos que estamos considerando mantienen, de manera manifiesta, los marcadores de su contexto y de sus condiciones de producción. Por lo tanto, aquí el texto no es sólo la objetivación secundaria y auxiliar de un conjunto de prácticas y acciones sociales previas. Es que esas prácticas y acciones consisten ya precisamente en buena parte en la producción de textos. Textos de todo tipo: conversaciones, informes, código fuente, bases de datos, *logs*, etc.

Así, en este caso no es posible entender sin más que

> la "textualización" [...] es el proceso a través del cual la conducta no escrita, el habla, las creencias, la tradición oral y el ritual son caracterizados como un corpus, como un conjunto potencialmente significativo separado de toda situación discursiva o performativa inmediata (Clifford, 1995, pág. 58).

Así caracterizaba Clifford el modelo interpretativo de Geertz, en el que se entendería la cultura como un conjunto de textos a interpretar, pero textos entendidos como fragmentos separados de sus condiciones de enunciación. Este modelo no sería aplicable en nuestro caso. La textualización de esos fragmentos de la forma de vida que intentamos entender y describir conserva gran parte de su contexto de producción e interacción, tal y como este contexto, al menos el que es común a los sujetos que se comunican, aparece ante los propios participantes. Son entonces los propios miembros de Debian, no el etnógrafo (aunque sí sea suya la selección y puesta en relación de lo que se incluye en la etnografía), quienes textualizan sus prácticas al tiempo que incorporan en este proceso buena parte de su horizonte interpretativo.

Es decir, en definitiva, esos fragmentos de los que estamos hablando se deben entender como algo que está entre el «texto» y el «discurso», según la distinción que Clifford recoge de Benveniste:

> el discurso [...] es un modo de comunicación en el cual la presencia del sujeto hablante y de la situación inmediata de la comunicación es

> intrínseca. El discurso está marcado por pronombres (pronunciados o implícitos), *yo* y *tú*, y por indicadores deícticos –*éste*, *aquél*, *ahora*, etcétera– que señalan la instancia presente del discurso más que algo que se encuentra más allá de él (Clifford, 1995, pág. 58).

Creo que es posible considerar que los recursos puestos en juego en la comunicación computacionalmente mediada, al dar necesaria e inmediatamente forma textual a los enunciados y las interacciones, recogen (quizás paradójicamente) buena parte de esos indicadores que son lo característico del discurso. Lo discursivo no se contrapone al texto, sino que lo incluye añadiéndole otras dimensiones. Esa forma textualizada es precisamente parte de, y producto de, las condiciones discursivas y dialógicas que constituyen la acción social. Pensemos por ejemplo en el fragmento de conversación de IRC que veíamos anteriormente, en el que el sujeto, el destinatario y el contexto de los enunciados, incluso el momento exacto de su enunciación, siguen siendo visibles mucho tiempo después. Tan presentes, al menos, como lo son en el momento mismo de la comunicación. No completamente, puesto que aun así no accedemos directamente al conocimiento o las expectativas compartidas, pero sí desde luego de una manera cualitativamente más completa que en el caso de una textualización *a posteriori* de una interacción oral, en la que el etnógrafo selecciona y transcribe los elementos que conserva. Todo esto lo iremos viendo en una gran cantidad de interacciones presentadas posteriormente, y veremos cómo este efecto se potencia por el uso de diferentes canales de comunicación, dispositivos de interacción y de colaboración simultáneamente. Todos estos indicadores permiten reconstruir después las cadenas de mediadores (y por tanto describir las cadenas de prácticas sociales) de los que hablamos en la sección siguiente.

Tampoco se pueden hacer suposiciones generales y *a priori* sobre las implicaciones del uso de estos medios y dispositivos tecnológicos. Sólo el trabajo de campo etnográfico podrá establecerlas. Partir de una distinción *a priori* entre formas de comunicación *online* y *offline* hará que la primera se entienda sólo a partir de sus carencias, como advertía Baym:

> If comparing mediated communications to face to face communication

> doesn't work adequately, it might be more fruitful to think of digital communication as a mixed modality that combines elements of communication practices in embodied conversation and in writing. Instead of approaching mediated interaction as face to face communication and finding it wanting, we draw from our existing repertoire of communication skills in other modes to make a medium do what we want it to do as best we can (Baym, 2010, pág. 63).

No se trata entonces de lamentarse porque una determinada forma de comunicación nos parezca extraña, sino de hacerse culturalmente competente en la evitación de las posibles dificultades de comprensión. Por otra parte, ésta es también la situación de cualquier miembro de Debian cuando empieza a colaborar.

Ardèvol *et al.* (2003, pág. 76) señalan también que una característica de la etnografía que toma como campo las interacciones *online*, es que está mediada tecnológicamente desde el principio hasta el final, de tal modo que, en la mayor parte de los casos, la observación es inseparable del registro. La mayor parte de las veces se observa un registro, o se produce un registro en el mismo hecho de observar. Este registro suele además estar disponible para cualquier interesado, abriendo así la etnografía a nuevas interpretaciones. Así, la específica mediación tecnológica que encontramos en investigaciones como la presente contribuye efectivamente a preservar el contexto y la posibilidad de interlocución de los diferentes agentes.

De ahí que aparezcan tantos enlaces a correos archivados públicamente, páginas *web* y otros documentos públicos en el presente trabajo. A veces señalan las fuentes usadas y analizadas, a veces permiten ampliar y profundizar en aspectos que aquí sólo se pueden sugerir, pero en cualquier caso permiten contrastar la interpretación que se ofrece.

1.3.2 Sobre la especificidad de la etnografía en contextos *online*.

Existen ya numerosas investigaciones etnográficas sobre lo que se ha llamado en muchas ocasiones «comunidades virtuales», y reflexiones metodológi-

cas sobre la etnografía virtual o realizada en contextos *online*. Me limitaré aquí a remitir a dos estudios bibliográficos muy completos sobre estas cuestiones, Coleman (2010a) y Wilson y Peterson (2002). En muchas de estas investigaciones y textos, un elemento central es la tematización de la especificidad de este contexto de prácticas de comunicación e interacción *online*, a partir de su contraposición con la acción social y la comunicación presenciales. En una investigación de estas características parece obligado poner el foco metodológico en estas aportaciones teóricas.

Sin embargo, lo específico del campo de mi investigación se trata metodológicamente mejor a partir de los conceptos (surgidos en el seno de la Teoría del Actor-Red) de trama y colectivo sociotécnicos que como un caso de (la posiblemente mal llamada) etnografía virtual. Ha sido el desarrollo de la propia investigación y sus necesidades lo que me ha ido llevando desde el énfasis en las características diferenciadoras de la acción social *online* y computacionalmente mediada al énfasis en la descripción de una red en el sentido de la Teoría del Actor-Red, es decir de una trama sociotécnica. En lugar de partir de una distinción *a priori* y engañosa, en lo que puede tener de esencializadora, entre lo *online* y lo *offline*, es mucho más productivo tratar la cuestión de la acción social computacionalmente mediada a partir del papel que en ella juegan los dispositivos tecnológicos y los actores (en términos de Latour) no humanos, siguiendo sus asociaciones. Un papel que, como mostraremos, es (precisamente) el de mediadores y no simples intermediarios. Como veremos, en tanto que transforman los procesos de traducción, transforman a los actores. La cuestión metodológica fundamental no está en lo *online* o lo «virtual», sino en la mediación tecnológica de la acción social.

El papel de los no humanos, de los dispositivos tecnológicos y su importancia en la acción social, nos lleva así, para poder describir las acciones sociales observadas, en la dirección de una antropología simétrica en el sentido de Latour (2007) y Callon (1984):

> Para que la antropología se vuelva simétrica [...] debe absorber lo que Michel Callon llama principio de simetría generalizada: el antropólogo debe situarse en el punto intermedio donde puede seguir a la vez

> la atribución de propiedades no humanas y humanas (Latour, 2007, pág. 143).

Es decir, no partir de un dualismo originario entre lo social y lo natural o técnico, ni del privilegio explicativo concedido en exclusiva a los actores humanos. Esta cuestión se verá reflejada especialmente en la controversia analizada sobre la distinción entre lo técnico y lo social (véase el capítulo 2, *Debian como trama sociotécnica*), así como en el papel que juegan los objetos híbridos (véase el capítulo 4, *El carácter híbrido del* software. Bugs, hackers*, paquetes y repositorios*). Es cierto que tanto Latour como Callon se refieren, para cuestionarla, a la relación naturaleza/sociedad, mientras que aquí hablaremos más bien de técnica que de naturaleza. Pero nos referimos en ambos casos a una dimensión que se ha constituido a partir de la perspectiva de una ciencia natural concebida tradicionalmente como desvinculada de todo componente social. En cualquier caso, lo fundamental son las asociaciones entre elementos heterogéneos, humanos y no humanos, que implican las ciencias naturales y la tecnología moderna:

> Las ciencias y las técnicas no son notables porque son verdaderas o eficaces [...], sino porque multiplican a los no humanos reclutados en la fábrica de los colectivos y vuelven más íntima la comunidad que formamos con esos seres (Latour, 2007, pág. 158).

Lo principal es entonces atender a las tramas de relaciones y asociaciones entre todos los actores implicados. La simetría generalizada lleva así a la noción de colectivo:[14]

> Ahora nos encontramos ante producciones de naturalezas-culturas que llamaré colectivos, para recordar bien que son tan diferentes de la sociedad de los sociólogos –los hombres-entre-ellos– como de la naturaleza de los epistemólogos: las cosas-en-sí (Latour, 2007, pág. 156).

[14] Véanse (Latour, 2007) y Latour (1996, cap. 6).

De ahí la importancia de las nociones de trama sociotécnica[15] (véase el capítulo 2, *Debian como trama sociotécnica*) y de colectivo (véase el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*). Si como hemos visto Debian nombra al mismo tiempo un Proyecto, una comunidad y una distribución, considerarlo como colectivo engloba toda esa complejidad. Reconstruir el trabajo de mediación de los híbridos, de los cuasi-objetos, seguirlos y tenerlos en cuenta, así como investigar el establecimiento de dualismos a partir de ellos (lo que Latour (2007) llama trabajo de purificación) es lo que permite entender el colectivo Debian.

Volviendo ahora a la distinción entre los contextos *online* y *offline*, ésta hay en todo caso, como decíamos, que construirla etnográficamente, en tanto que los problemas y controversias que aparecen relacionados con esta distinción son, en mi campo, específicos de la creación y distribución de *software* libre y de servicios e infraestructuras para ello. No hay ni puede haber, como iremos mostrando, una división nítida de contextos de interacción mediados tecnológicamente por un lado y presenciales por otro. Hay una continuidad no sólo entre ellos, sino entre cualquier forma de interacción que se da en Debian y los procesos de creación y distribución de *software* libre.

Así, aunque he tenido que realizar mucha investigación empírica y mucha observación a través de medios digitales, el problema metodológico fundamental que se ha planteado no es el de las limitaciones que esos medios puedan imponer, sino el de describir las mediaciones que se producen. No se trata de reflejar un nuevo tipo de espacio (*online* o «virtual») por oposición al espacio *offline* y presencial, sino de describir las mediaciones que introducen determinados dispositivos tecnológicos. Son éstas las que requieren una descripción. En una red, en una trama, por definición, no hay contexto que contenga la realidad que queremos conocer: las conexiones, asociaciones y cadenas de mediaciones nos llevarán siempre más allá.

Así pues, como señalan Miller y Slater (2000, pág. 6), «Rather than starting from virtuality, then, we are concerned to start our investigation from within the

[15] Un concepto más adecuado que el de «red», en tanto alude más directamente a la trama etnográfica como objeto de descripción.

complex ethnographic experience». Así hemos intentado proceder, independientemente de que esta experiencia etnográfica sea presencial o mediada computacionalmente. Hemos partido en la escritura de esta etnografía de prácticas concretas, siguiendo sus conexiones y asociaciones, para describir la red que trazan. Y así se vinculan interacciones *online* y *offline*. Las asociaciones y mediaciones se encuentran antes, en la experiencia etnográfica concreta, que las interacciones cara a cara o el contexto que las explicaría. Tiene razón Latour cuando afirma:

> If context was an impossible starting point, so are face-to-face interactions. [...] In humans more so than in monkeys, interference, dispatching, delegation, and articulation are visible and should offer us, in place of local face-to-face interactions, an excellent point of departure (Latour, 2005, pág. 1991).

En la investigación etnográfica sobre Debian ocurre exactamente eso. Aunque se intente partir de las interacciones entre individuos, desde el primer momento nos vemos llevados a introducir consideraciones y explicaciones sobre la interacción de aquéllos con todo tipo de objetos y dispositivos, y de éstos entre sí. Describir las prácticas de los sujetos y su forma de vida requiere hablar de cómo funciona el IRC, o de qué es un paquete de *software*, o cómo funciona una licencia o un repositorio de *software*. Y no sólo describirlas: también empezar a interrogarse por ellas. De hecho, parte del interés original que me llevó a estudiar las comunidades del *software* libre era también un interés por entender el *software* libre mismo, y no tan sólo una serie más o menos novedosa de interacciones y prácticas sociales. En cualquier caso, se trata de partir de las mediaciones: si la descripción de las prácticas sociales que se dan en torno al *software* libre nos lleva inevitablemente a intentar entender cómo funciona ese *software*, también es cierto que partir del intento de comprensión del mismo nos lleva de manera igualmente inevitable a la descripción de dinámicas y prácticas sociales de todo tipo.

Esto es lo que iremos haciendo en los diferentes capítulos, introduciendo la descripción de esas mediaciones y de los dispositivos tecnológicos según sea necesario para dar cuenta de la acción social considerada en cada caso. La cuestión

específica de los medios de comunicación y colaboración *online* se retomará en el capítulo 4, *El carácter híbrido del* software. Bugs, hackers, *paquetes y repositorios* y en la sección 6.2, *Interacción entre aspectos* online *y* offline. *DebConfs*.

Irá así quedando claro que los procesos sociales que analizaremos no tienen lugar en una dimensión «virtual» de interacción desconectada de lo que constituye el continuo de interacciones cotidianas de sus participantes. Lo que ocurre en esas cadenas de acción nos lleva más allá de sí mismas, conectándolas necesariamente con otros contextos, otras situaciones sociales y otros objetos. Difícilmente una interacción cara a cara en un contexto presencial poseerá una mayor densidad de supuestos e implicaciones, cuyo seguimiento y descripción constituyen la condición de posibilidad del trabajo etnográfico, que muchas de las interacciones complejas a través de medios digitales que iremos tratando. Por supuesto que tenemos que atender a la especificidad de estas formas de interacción y de sus condicionantes tecnológicos, pero no son éstos en ningún caso los que determinan la forma social de la interacción. Tampoco son medios transparentes, transmisores de una información en la que no influyen. La experiencia etnográfica de intentar seguir y describir este tipo de interacciones complejas atestigua su riqueza, que no se agota en el espacio «virtual» delimitado y configurado por las características tecnológicas propias de los medios de comunicación electrónica.

De hecho, lo que esta multiplicación de interacciones electrónicas nos permite es *localizar* de manera compleja la información obtenida, según el contexto concreto en el que se produce, y *triangularla*, al localizarla en diferentes fuentes.[16] En definitiva, construir una descripción densa:

> Cuando el relato de unos hechos ordinarios y concretos *condensa* una visión relacional de valores y significados culturales, compuesta por el investigador, estamos ante una descripción densa. [...] la descripción densa [...] exige que captemos, con el mayor detalle y alcance posibles, el proceso por el que esos significados e intenciones acaban construyendo un espacio público, es decir, común, de sentidos y valores compartidos o negociables. Reflejar ese espacio público dando cuenta

[16] Véase Velasco y Díaz de Rada (2004, págs. 220-223).

> de sus múltiples niveles y dimensiones es el objetivo de la descripción densa (Velasco y Díaz de Rada, 2004, pág. 220).

Lejos de conducirnos a una etnografía deslocalizada y ocupada en una construcción subjetiva forjada en la pura voluntad desterritorializada de los agentes sociales, la atención a las conexiones que se producen en la comunicación mediada computacionalmente nos permite la posibilidad de realizar una descripción densa. Es posible ofrecer una descripción densa de este espacio social porque, por un lado, los contextos de interacción son en sí mismos complejos, creando un espacio que, independientemente de que sea *online*, constituye una red, una trama formada por múltiples conexiones; y porque la experiencia de intentar reconstruirla es en sí misma compleja, y nos lleva necesariamente hacia lugares que no habíamos previsto.

Un último comentario sobre la cuestión del *holismo* en antropología. Por un lado, una de las conclusiones de Marcus (1995) era precisamente que no es posible una representación holista de la totalidad. Por otro lado, «la etnografía no puede elegir ser o no holista: será holista o no será nada» (Cruces, 2003, pág. 168). Lo que se hace necesario ante este dilema es encontrar formas de hallar las conexiones entre las diferentes manifestaciones de las prácticas culturales, de articular el contexto. Perseguir el holismo de las conexiones, no el de la totalidad cerrada. Si entendemos el holismo como el intento de crear un determinado tipo de significado, caracterizado por la intención de extender el contexto hasta donde nos lleven los fenómenos y las interacciones observadas, podemos renunciar a una visión omniabarcadora, a la pretensión de dibujar un mapa que represente la realidad, en favor de la descripción de los caminos y conexiones que sean significativos, de una trama en este caso sociotécnica.

1.4 Avance de los capítulos.

Retomando y resumiendo algunos de los elementos que han ido apareciendo hasta ahora, se pueden señalar dos grandes objetivos de la presente investigación. Por una parte, se trata de entender la relación entre el fenómeno del don (en este caso del código fuente, con todo lo que hemos visto que eso implica) y la construc-

ción de una comunidad constituida en torno a una actividad muy técnica pero con fuertes implicaciones éticas y políticas. De aquí la primera parte del título de esta tesis, *Del* Hau *a la GPL*. No se trata aquí (como veremos en el capítulo 5, *El don del* software *libre*) de una operación análoga a la realizada por Lévi-Strauss, desde el *hau* a las estructuras. Tampoco se trata de seguir la línea de Godelier, para quien la sustancia del don serían las relaciones sociales entre los seres humanos. La GPL no es una estructura ni una sustancia, sino que refiere como artefacto técnico-legal a todas sus relaciones y vínculos, a toda la trama sociotécnica en la que se inserta.

Lo que nos lleva al segundo gran objetivo y a la segunda parte del título: *Una etnografía de la constitución de Debian como colectivo sociotécnico*. Caracterizar Debian, en tanto objeto etnográfico, como un colectivo sociotécnico supone comprender las prácticas y vínculos sociales estudiados en relación a los dispositivos tecnológicos y los procesos de mediación técnica, a los que se da una papel muy importante en esta investigación. De lo que se trata es de describir un entramado sociotécnico.

A eso se dedica fundamentalmente el capítulo 2, *Debian como trama sociotécnica*. Se empieza describiendo en detalle un proceso social tecnológicamente complejo, que se produjo en forma de controversia, y todo el conjunto de acción que se despliega en torno a ese proceso. Para darle sentido se introducirán muchos conceptos que contribuyen a la comprensión de cómo se conforma la vida social de Debian. Esta descripción permitirá entender qué tipos de agentes se hace necesario tener en cuenta para describir los procesos de traducción que constituyen la trama sociotécnica objeto de la presente investigación, y será un punto de referencia a lo largo de todo este trabajo. La segunda parte del capítulo examina la construcción de la distinción, en el seno de Debian, entre lo técnico y lo social. Más que de simplemente eliminar este dualismo, de lo que se trata es de describir etnográficamente su producción, cómo se construye. Se trata pues de examinar la construcción de ese dualismo, pero no de establecer la «construcción social» del mismo. No se trata de que sea un artefacto establecido a partir de un cierto contexto social, sino que al contrario, una parte de lo que se construye es precisamente «lo social» (Hacking, 1999; Latour, 2005). Se establecen así diferentes sentidos, *emic* y *etic*, de esos conceptos, y su papel en la vida social de Debian.

La última sección sobre agencia y semiótica permite acabar de perfilar el sentido en el que lo técnico y lo social son realidades en proceso, objetos que se están creando y recreando a partir de la acción social de los implicados. También permite profundizar en el carácter distribuido de la agencia.

Si el capítulo 3, *Formas de gobierno. Doocracia, agencia y apertura*, nos devuelve a algunos tópicos (sólo) aparentemente más tradicionalmente propios de la etnografía, este paso refleja también la experiencia típica del miembro de Debian, que primero se suele ocupar de un trabajo más técnico, y después se implica más en las formas de organización y gobierno.

Este capítulo se divide claramente en dos secciones, dedicados a los dos conceptos que considero más importantes para entender la organización de Debian y los procesos de toma de decisiones, ambos en relación a los resultados del capítulo anterior relativos al papel de los dispositivos tecnológicos. La primera de ellas se organiza en torno al concepto de doocracia, que consiste básicamente en la forma de gobierno en la que quien puede tomar una decisión es quien lleva a cabo la actividad, y se articula con los de meritocracia, consenso y democracia. Este concepto permite mostrar el carácter procesual y producido del poder, de la capacidad de decisión y de la agencia. Es inseparable del carácter distribuido de la agencia, y tiene una fuerte vinculación con el don y la obligación de recibir, y con el proceso de entrada en la comunidad. En definitiva, la doocracia no supone sólo una forma de gobierno, sino una forma de constituir actores a partir de la mediación de sistemas tecnológicos. También se presta mucha atención a las tensiones que produce en torno a la concentración de poder.

La otra sección describe a Debian también como un público recursivo, y se establecen las relaciones de este concepto con el colectivo Debian, a partir de la radical modificabilidad técnica y moral de su infraestructura. Veremos en qué sentido depende de la apertura de sus procesos y sus prácticas sociales. Por último, se establece la relación entre doocracia y público recursivo.

En el capítulo 4, *El carácter híbrido del* software. Bugs, hackers*, paquetes y repositorios*, se trata de examinar más detenida y específicamente los dispositivos tecnológicos que dan consistencia a Debian como trama sociotécnica y como

colectivo. Es a partir de éstos que se produce la separación entre lo técnico y lo social, al tiempo que posibilitan la doocracia a partir del establecimiento de relaciones con ellos. Además, son los objetos que se donan en este campo. Estamos pues en un capítulo central.

Tras unas consideraciones más teóricas sobre la naturaleza del *software* y del código, a partir de conceptos como los de agenciamiento, actante, mediadores u objetos híbridos (que permiten deshacer toda una serie de dualismos), se ofrece una descripción de la práctica social que define a Debian, el empaquetado de *software* libre. Pasamos a partir de ahí a considerar el papel de Debian como distribución y su relación con el entorno del *software* libre en general.

El capítulo 5, *El don del* software *libre*, trata de iluminar algunos aspectos de nuestro campo de investigación a partir de las teorías antropológicas clásicas sobre el don, especialmente a partir de las aportaciones de Weiner y Godelier. Pero no se trata en ningún caso de entender el fenómeno del *software* libre como un ejemplo de las llamadas economías del don. Veremos que aquí sigue siendo central el papel concedido a los objetos técnicos y su circulación en la formación de vínculos sociales.

El hilo conductor serán las tres obligaciones respecto al don que distinguía Marcel Mauss. Respecto a la primera, la obligación de dar, se profundiza en las características del código fuente (el objeto que circula como don) y se discuten algunas de las motivaciones de los miembros de Debian para contribuir al Proyecto. Se mostrará entonces en qué sentido el *software* libre puede ser considerado simultáneamente como don y como posesión inalienable, y como una manera de disolver la paradoja, expuesta por Weiner y desarrollada por Godelier, de «guardar para (poder) donar y donar para (poder) guardar». Al hablar de la segunda obligación, la de devolver, se examina el lugar central de las licencias libres, como el mecanismo fundamental que mantiene los bienes en circulación, en el colectivo sociotécnico, y se introduce el concepto fundamental de procomún en la descripción de las prácticas sociales en Debian. Se discute aquí también la cuarta obligación del don, en relación con el concepto de gracia y el lugar estructural del procomún, y se vincula el funcionamiento del procomún con el concepto de «posesiones inalienables». La siguiente parte intenta determinar hasta qué punto

se puede hablar en nuestro campo de la tercera obligación, la de recibir (que ha sido de hecho la menos tratada históricamente), y en qué prácticas y dispositivos se encarna. Veremos que está muy estrechamente ligada a la apertura propia del Proyecto Debian. Por último, se relaciona todo lo expuesto con el concepto de doocracia (como un sistema en el que se crea valor y significado a partir de la acción) y con la formación del colectivo sociotécnico.

El capítulo 6 y último, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*, se centra en la aplicación de los conceptos de comunidad y colectivo a Debian, y se describen toda una serie de controversias que se han producido a lo largo de su historia respecto a la cuestión de quién es miembro de Debian y cómo se llega a serlo. Al hablar de colectivo sociotécnico, se enfatizan los componentes tecnológicos del colectivo, pero es fundamental el hecho de que éste se constituye en torno al don. En esa relación juega un papel fundamental el concepto de comunidad. Se trata en este caso de una comunidad constituida en torno a una forma de don y mediada por objetos y dispositivos técnicos.

Tras reconstruir el sentido *emic* de la comunidad en Debian, la parte central de este capítulo investiga el proceso de ingreso y las condiciones de pertenencia al Proyecto Debian. La hipótesis fundamental en esta parte es que es precisamente esa concepción *emic*, y la constitución del colectivo en torno al don del *software*, lo que lleva a las diferentes reformas y controversias que se han producido en Debian respecto a quién debe ser considerado como miembro. Aquí será esencial un análisis detallado del *New Member Process*. Y es que, si está claro quién forma parte del Proyecto Debian, no lo está tanto quién forma parte de la comunidad.

En este capítulo se tratan también dos cuestiones más específicas relacionadas con la comunidad y el colectivo: la interacción de las formas de comunicación a través de medios electrónicos, y las presenciales; y el papel de la identidad (asegurada tecnológicamente) y la confianza. Ambos aspectos dependen de la relación sociotécnica entre individuos y dispositivos tecnológicos, y podrían haber sido igualmente tratados en algún capítulo anterior, pero su relación con las ideas de construcción de la comunidad y colectivo hace que el lugar más adecuado para tratarlos sea éste.

Hemos pasado así de la imbricación entre aspectos tecnológicos y sociales al significado y la agencia, de las formas de organización social y las formas de colaboración a las características del código, del don a la comunidad, y de ésta de nuevo a los objetos tecnológicos. Los temas tratados están relacionados de tal modo que cualquiera de estos aspectos nos lleva necesariamente a los demás. Esto se ve intensificado por la necesidad de explicar con cierto detalle las características técnicas de muchos de los dispositivos y objetos implicados, lo que requiere a su vez ir adelantando muchas de las cuestiones que se verán posteriormente. Esto supone que el orden de exposición seguido tiene algo de arbitrario, pero no más que otras alternativas.

Por último, se ofrecen al final unas muy breves conclusiones, o más bien unas reflexiones generales sobre algunos de los temas que para entonces habrán aparecido.

2 Debian como trama sociotécnica.

> *If the social circulates and is visible only when it shines through the concatenations of mediators, then this is what has to be replicated, cultivated, elicited, and expressed by our textual accounts. The task is to* deploy *actors* as *networks of mediations –hence the hyphen in the composite word 'actor-network'.*
>
> —Latour (2005, pág. 136).

En este capítulo trato de ofrecer una descripción etnográfica de un conjunto de prácticas y de procesos de formación de vínculos sociales que se producen en el seno del Proyecto Debian. En definitiva, de la forma o formas de vida que surgen al participar en esas prácticas y vínculos.

Pero si a primera vista algo caracteriza a estas prácticas y los vínculos que crean, es la omnipresencia de elementos mediadores que podríamos llamar dispositivos tecnológicos. La centralidad de la mediación técnica en absoluto está limitada a conjuntos de acción organizados en torno el desarrollo tecnológico. Me limito a señalar que este lugar central es aquí no sólo más claro, sino ineludible. Y es que estamos frente a una forma de vida fuertemente marcada por la tecnología y su desarrollo. Si, como hemos apuntado en la introducción y profundizaremos en la sección 4.2.2, *Mediadores e intermediarios. Las figuras del* bug *y del* hacker, la del *hacker* es una figura que expresa mucho del *ethos* o el *habitus* de los Desarrolladores de Debian, se puede caracterizar su práctica en este ámbito a partir de la definición de *hacker* que me dio un miembro de Debian: «**dato:** un *hacker* es alguien que tiene una relación especial con la tecnología ... o mejor

dicho, una relación con el mundo a través de la tecnología». Aquí, en este momento de vacilación y corrección posterior, se muestra con claridad lo esencial del asunto que trato de mostrar: no se trata tan sólo de entender las prácticas estudiadas *en relación externa* al uso de tecnologías sofisticadas (que para el caso podrían ser esas u otras muy distintas), sino de comprenderlas como construcción de mediaciones entre elementos heterogéneos, muchos de ellos tecnológicamente sofisticados. Construcción de un mundo, una forma de vida o, como veremos, de un colectivo sociotécnico.

En definitiva, debemos evitar contentarnos con entender ese conjunto de dispositivos y artefactos como intermediarios (meros instrumentos de una agencia que les sería completamente externa),[1] estableciendo una división entre relaciones o vínculos entre personas, y el papel inerte de esos dispositivos. Nos veríamos así limitados a investigar una dinámica social independizada de aquello que le proporciona consistencia y estabilidad. En el caso de Debian sería entonces imposible comprender simplemente que siga existiendo después de tantos años. Lo interesante ocurre precisamente en esos procesos de mediación y traducción que tienen lugar en las cadenas de prácticas que vamos a describir.

Así, Debian aparece como un campo privilegiado para estudiar la formación de colectivos (caracterizados, en la perspectiva de Latour, por las asociaciones entre humanos y no humanos) sociotécnicos:[2] los dos problemas que está permanentemente tratando de resolver son la integración de nuevos individuos y la integración de dispositivos técnicos, fundamentalmente programas. De hecho, hasta hace poco (como veremos), ambas cuestiones se unían en un solo proceso: la condición para llegar a ser miembro de Debian de pleno derecho era integrar algún paquete de *software* en la distribución. Comprender el conjunto de acción y cadenas de prácticas que sostienen ese colectivo requiere incluir ya en la descripción etnográfica, como elementos relevantes en su especificidad, todos los elementos que lo configuran. Partimos pues de un conjunto de elementos heterogéneos que construyen el colectivo.

[1] Esta distinción entre *mediadores* e *intermediarios* (Latour, 2005) la desarrollaré en la sección 4.2.2, *Mediadores e intermediarios. Las figuras del* bug *y del* hacker.

[2] Esta cuestión se desarrolla en el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*

Hablar de Debian como entramado sociotécnico significa entonces dar importancia desde el principio a los aspectos tecnológicos que contribuyen a su configuración. El término entramado hace referencia a la trama sociocultural como tejido de relaciones, significados y prácticas, pero también a la trama del relato etnográfico como despliegue de esas relaciones en la escritura etnográfica, en el sentido expuesto por Velasco y Díaz de Rada (2004, capítulo 6). A fin de cuentas, ese es también el espíritu de la cita que abre este capítulo.

En el estudio de una forma de vida, como es el caso, caracterizada por una íntima relación con la producción y distribución de tecnología, no podemos partir *a priori* de una distinción entre los procesos sociales, que serían aquellos que nos interesan como antropólogos, y los procesos «puramente técnicos» que quedarían más allá, o más acá, de nuestra posibilidad de comprensión como científicos sociales. Hacer eso nos condenaría a no entender la especificidad de esos procesos, ni la manera en que la tecnología contribuye efectivamente a la configuración de una forma de vida como la que estamos considerando (en definitiva, de cualquier forma de vida humana). O, más que contribuir, cómo forma parte integrante y efectiva de ella.

El campo de los Estudios Sociales sobre la Ciencia y la Tecnología nos permite poner en cuestión esta dicotomía. Dentro de éstos, la perspectiva de la Construcción Social de la Tecnología nos podría ofrecer una primera aproximación. En su modelo de evolución tecnológica (Pinch y Bijker, 1989), los elementos esenciales no son sólo los problemas técnicos y sus posibles soluciones, sino que juegan un papel fundamental los grupos sociales implicados, y el significado que estos dan a los artefactos en cuestión. Los factores sociales no son sólo un factor externo al proceso de desarrollo tecnológico. Las interpretaciones que los distintos grupos sociales asignan a las diferentes soluciones condicionan también la dirección del desarrollo tecnológico.

O, como leemos en otro artículo de esa misma obra:

> Technological systems contain messy, complex, problem solving components. They are both socially constructed and society shaping. Among

> the components in technological systems are physical artifacts, such as the turbogenerators, transformers, and transmission lines in electric light and power systems. Technological systems also include organizations, such as manufacturing firms, utility companies, and investment banks, and they incorporate components usually labeled scientific, such as books, articles, and university teaching and research programs. Legislative artifacts, such as regulatory laws, can also be part of technological systems. Because they are socially constructed and adapted in order to function in systems, natural resources, such as coal mines, also qualify as system artifacts (Hughes, 1989, pág. 51).

No sería inadecuado considerar al *software* libre en general, y al Proyecto Debian como una parte integrante del mismo, como un «sistema tecnológico» en este sentido. Y si Hughes tiene razón, los diferentes componentes (artefactos físicos, organizaciones, científicos, legislativos, recursos naturales; iremos añadiendo algunos más, adecuados para entender el funcionamiento de los proyectos dedicados al *software* libre) de un tal sistema tecnológico son tanto «socialmente construidos» como «configuradores de la sociedad». Y no tiene sentido, por lo tanto, suponer que alguno de ellos tiene más importancia para el científico social. Ni que otros se desarrollan según una lógica independiente de los contextos de acción en los que los encontramos:

> Because organizational components, conventionally labeled social, are system-builder creations, or artifacts, in a technological system, the convention of designating social factors as the environment, or context, of a technological system should be avoided (Hughes, 1989, pág. 52).

En las secciones siguientes de este capítulo se mostrará precisamente cómo la distinción entre «factores sociales» y sistema tecnológico es precisamente eso, una convención y una producción. Una distinción que es producida en la práctica social. Pero es la perspectiva de la Teoría del Actor-Red de Bruno Latour y Michel Callon la que nos permitirá poner más radicalmente en cuestión este dualismo, presente todavía en la concepción de Hughes. Y sobre todo, es mucho más cercana a la empresa etnográfica. En particular, nos permitirá evitar la tentación de

intentar explicar los fenómenos en función de determinadas «fuerzas sociales» que actúan condicionando a los individuos sin tener en cuenta la dinámica propia de los objetos tecnológicos que forman parte integrante del campo que nos ocupa:

> As soon as you believe social aggregates can hold their own being propped up by 'social forces', then objects vanish from view and the magical and tautological force of society is enough to hold every thing with, literally, no thing (Latour, 2005, pág. 70).

> [...] any thing that does modify a estate of affairs by making a difference is an actor –or, if it has no figuration yet, an actant (Latour, 2005, pág. 71).

Los dispositivos tecnológicos no sólo son actantes (en la terminología de la Teoría del Actor-Red, el término actante, tomado de la semiótica, no significa más que un actor, un agente, sin el presupuesto de que sea un ser humano) en tanto que efectivamente producen una diferencia en la configuración de la realidad, sino que son fundamentales para entender las dinámicas sociales que producen relaciones de poder (como veremos en el capítulo 3, *Formas de gobierno. Doocracia, agencia y apertura*):

> what is new is that objects are suddenly highlighted not only as being full-blown actors, but also as what explains the contrasted landscape we started with, the overarching powers of society, the huge asymmetries, the crushing exercise of power (Latour, 2005, pág. 72).

Es decir, hemos de tomar seriamente en consideración la agencia de los objetos tecnológicos para entender cómo se configuran los fenómenos que queremos analizar y comprender. Entender la acción en cualquier campo requiere anular la distinción entre dos ámbitos que siguen lógicas independientes, sin afectarse más que externamente:

> ANT [Actor-Network Theory] states that if we wish to be a bit more realistic about social ties than 'reasonable' sociologists, then we have to accept that the continuity of any course of action will rarely consist of human-to-human connections (for which the basic social skills would be enough anyway) or of object-object connections, but will probably zigzag from one to the other (Latour, 2005, pág. 75).

En este capítulo me centraré en un conjunto de prácticas más marcadamente tecnológicas, en las que la descripción necesitará una atención detallada al uso de dispositivos y sistemas técnicos. Pero no debemos olvidar que estamos tratando con una forma de vida caracterizada justamente por eso, por centrarse en prácticas tecnológicas sofisticadas y complejas. Aquí, las prácticas sociales son casi siempre y de manera profunda prácticas de mediación con y mediante la tecnología. Ciertamente, estas prácticas técnicas, aun cuando no se dan en un contexto de interacción social cara a cara, están creando vínculos sociales concretos y configurando profundamente una forma de vida social.

2.1 Bug #573745.

Empecemos pues la descripción con una controversia que seguiré desplegando en las secciones posteriores. El 5 de octubre de 2012 se mandó el siguiente correo electrónico a la lista de correo `debian-devel-announce` (Debian-devel-announce, 2012a):[3]

```
From: Don Armstrong <don@debian.org>
Subject: [CTTE #573745] Maintainership of python packages in Debian
To: debian-devel-announce@lists.debian.org
Date: Fri, 5 Oct 2012 13:53:18 -0700
Mail-Followup-To: debian-ctte@lists.debian.org
Resent-From: debian-devel-announce@lists.debian.org
PGP Signed Part:Good signature from A2D13001D98C0FBA Don Armstrong <don@debian.org>
(trust full) created at 2012-10-05T22:53:13+0200 using RSA

== RESOLUTION ==
```

[3] Lo que se describe en este capítulo es válido en la época de la resolución del *bug*. Algunos detalles sobre el empaquetado de Python o el funcionamiento y composición del Comité Técnico han podido variar desde entonces.

The technical committee was asked in #573745 to consider replacing the current maintainer of python (Matthias Klose.) Multiple issues were presented at the time, including a lack of communication from the python maintainer, delays in uploads of new versions of python to unstable/experimental, and a lack of coordination with packaging helpers such as python-support, and python-central.

1. The communication between the python maintainer and other individuals affected by python packaging has not been ideal. These breakdowns appear to be rooted in an unfortunate feedback loop, of which all parties involved share some blame.
   a) On multiple occasions, inflammatory comments regarding the employment and/or motives of individuals involved in python have been made.
   b) The target(s) of these inflammatory comments then decline to respond to any messages from the offending parties, and also reduce public communication to other parties lest further hurtful and demotivating comments ensue.
   c) The lack of response is taken as further confirmation of motives/bias, and decreases the threshold for additional terse or inflammatory comments. This reinforces b, completing the loop.

Neither the inflammatory comments, nor the lack of response are acceptable outcomes.

2. No mediation was attempted by a party respected by the involved parties until the pattern was well established and very difficult to overcome. In the future, everyone would be better served if similar issues were resolved in a nascent stage.
3. To resolve this issue, the committee has two options. It can either replace the maintainer, or it can decide not to replace the maintainer. Either decision will appear to validate one problematic behavior or the other.
4. All relevant and interested parties have been canvassed, via the -python list, to find what the possible teams of maintainers are for the python interpreter packages. See https://lists.debian.org/msgid-search/20120403083658.GB30420@upsilon.cc and the ensuing thread.

Therefore,

5. The committee expresses its disappointment in the communication problems which have lead to this issue, and strongly suggests that all involved parties be as awesome to each other as possible. In the advent of communication failures or problems, we request that any involved party contact a third party (such as a member of the technical committee) to mediate.
6. The committee requests that all major changes in the python interpreter packages which will affect other packages in Debian be announced on

```
     the appropriate mailing lists before they take effect so they can
     be planned for and/or unplanned problems discussed.
  7. The committee declines to change the maintainer of the python interpreter
     packages in Debian.
  8. The committee requests that Matthias Klose consider adding additional
     co- maintainers to the python interpreter package.
== END OF RESOLUTION ==
Please see http://bugs.debian.org/573745 for discussion of this bug.
- -
This space left blank intentionally.
End of PGP Signed Part
```

Con este correo electrónico se ponía fin, al menos momentáneamente, a un largo proceso de disputas en torno a una cuestión compleja para el profano en lenguajes de programación y el desarrollo de una distribución de *software* libre: el mantenimiento de `python` en Debian. Ha sido un proceso largo, que se remonta al menos a la Conferencia Internacional de Debian celebrada en México en 2006, y en ocasiones áspero, mostrando en sus momentos de máxima tensión la cultura de confrontación en la comunicación que ha conformado algunos de los conflictos que se han producido en Debian. Pero también muestra cómo ha ido cambiando a lo largo de los años, así como las posibilidades y dificultades del trabajo colaborativo voluntario. En las entrevistas ha sido uno de los temas recurrentes, y su seguimiento me ha permitido entender muchos matices de los procesos y las prácticas sociales que dan forma al Proyecto Debian. Esto último precisamente por la necesidad de entender no sólo los conflictos personales o de coordinación, sino también los artefactos tecnológicos implicados en todo este conjunto de acción.

Lo que tenemos aquí es un mensaje a una lista de correo (véase el cuadro 2.1, *Listas de Correo*). En concreto, la lista a la que se manda, `debian-devel-announce`, se usa para hacer anuncios de interés para los Desarrolladores de Debian. Aunque cualquiera puede leerla y suscribirse, sólo pueden mandar mensajes quienes ya son Desarrolladores, y no se crean hilos de discusión en la lista. En todo caso, se pueden mandar mensajes en relación a un anuncio a las listas `debian-devel` o `debian-project`. La primera de ellas se utiliza para discutir desarrollos técnicos relativos al Proyecto, la segunda para discutir asuntos no técnicos. Ambas son públicas y cualquiera puede escribir un mensaje o participar en un hilo de discusión. En este caso, si nos fijamos en la cabecera del correo, la lista a la que se

Las **Listas de Correo** permiten, a través del correo electrónico, el envío de mensajes a todos los suscriptores. El mensaje se manda a la dirección de la lista, p. ej., `debian-project@lists.debian.org`, y es automáticamente reenviado a todos los suscriptores de la lista. De la página de Debian dedicada a sus listas de correo (Debian.org, s.f.[h]):

> Debian GNU/Linux is developed through distributed development all around the world. Therefore, e-mail is the preferred way to discuss various items. Much of the conversation between Debian developers and users is managed through several mailing lists.
>
> There are many world-open mailing lists, meaning anyone can read everything that is posted, and participate in the discussions. Everyone is encouraged to help development of Debian and to spread the word of free software. There are also a few lists which are only open to official Debian developers; please don't interpret this as closed development, it sometimes doesn't make much sense discussing internal topics with non-developers.
>
> [...]
>
> The mailing lists are public forums.
>
> All emails sent to the lists are distributed both to the list subscribers and copied to the public archive, for people to browse or search without the need to be subscribed.

Cuadro 2.1: Listas de Correo

manda es `debian-devel-announce@lists.debian.org`, pero si se contesta al mismo la respuesta será mandada a la lista `debian-ctte@lists.debian.org` (según indica la cabecera `Mail-Followup-To:`). Esta ha sido una situación corriente durante el trabajo de campo: observar una práctica social concreta requiere atender simultáneamente a varias formas de comunicación. Lo veremos más adelante en este caso, y de forma más general en la sección 6.2, *Interacción entre aspectos* online *y* offline. *DebConfs*.

Este mensaje comunica pues la resolución tomada por el Comité Técnico de Debian en torno a una disputa sobre el mantenimiento del paquete `python` en la distribución. Para entender todo esto, es preciso explicar en qué consisten estos

elementos. Esta explicación se limitará, en esta sección, a lo esencial para poder entender el proceso social que se describe. Pero se irá ampliando y profundizando en las secciones siguientes.

2.1.1 Python como paquete.

En primer lugar, nos detendremos en qué es un paquete de *software* y qué significa mantener un paquete. En este contexto, un paquete es el conjunto de archivos necesarios para implementar alguna funcionalidad o característica en un sistema informático (véase Debian.org/doc (s.f.[b]); se ampliarán estas cuestiones en 4.5, *El empaquetado como práctica sociotécnica*). Normalmente se corresponderá con algún programa, pero puede ser también una biblioteca (un conjunto de funciones implementadas en un determinado lenguaje de programación que pueden ser usadas por diferentes programas), una fuente tipográfica, documentación, imágenes, etc. Es la unidad mínima de *software* que se puede instalar o desinstalar en un sistema como parte de la distribución, usando sus herramientas.

En Debian existen dos tipos de paquetes: fuentes y binarios. Un paquete binario es un archivo que contiene todo lo necesario para instalar un programa o un componente en el sistema. Un paquete fuente es la fuente (inteligible para un ser humano) a partir de la que se construye el paquete binario (inteligible por un computador). El *software* que es empaquetado para Debian normalmente no ha sido escrito específicamente para Debian (cuando sí lo ha sido, se llama *paquete nativo*), sino que ha sido desarrollado independientemente (se llama *upstream* a sus desarrolladores) en algún otro lugar, y el trabajo de empaquetarlo consiste en hacer lo necesario para integrarlo en la distribución. Debian como distribución es fundamentalmente una colección de paquetes integrados. Nos detendremos detalladamente en estos conceptos y procesos y sus diferentes dimensiones en el capítulo 4, *El carácter híbrido del* software. Bugs, hackers*, paquetes y repositorios*.

`Python` (Python.org, s.f.) es un lenguaje de programación ampliamente usado.[4] Es un lenguaje de alto nivel, interpretado, multiplataforma, licenciado co-

[4] No existe una forma clara de medir la popularidad de un lenguaje de programación, pero se publican algunos índices basados en diferentes criterios. En dos muy usados, `python` aparece en el puesto 8 (Tiobe.com, s.f.) y en el 4 (Pypl.github.io, s.f.). Ambas cifras corresponden a mayo de

mo *software* libre, y es compatible con diversos paradigmas de programación. En Debian (Wiki.debian.org, s.f.[d]) se encuentran no solamente varias versiones del lenguaje, a veces incompatibles, sino numerosos módulos (extensiones del lenguaje) y programas escritos en él, que a su vez puede que dependan de una versión específica del intérprete para funcionar. El intérprete de `python` es el programa que ejecuta a su vez el *software* escrito en el lenguaje `python`. Además del intérprete existen numerosas bibliotecas que extienden la funcionalidad del lenguaje. Así pues, el mantenimiento de `python` es complejo, al tiempo que fundamental para muchos otros programas escritos en ese lenguaje.

El mantenedor de un paquete es el encargado de construirlo a partir del *software* desarrollado por el desarrollador o desarrolladores *upstream*, integrarlo en la distribución, y mantenerlo libre de *bugs* (fallos). Existen diferentes figuras de miembros que pueden ser mantenedores, como veremos en el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?* En Debian, el mantenimiento de la mayoría de los paquetes relacionados con `python` se hacía en el seno de tres equipos principales: *pkg-python*, el *Debian Python Modules Team*, y el *Python Applications Packaging Team*. Pero el mantenimiento del propio intérprete de `python` y sus diferentes versiones lo realizaba un desarrollador individual, Mathias Klose (con el *nickname* de *doko*). La práctica de empaquetar *software* para Debian puede llegar a ser realmente compleja. Para automatizar y facilitar este trabajo se usan otros programas específicos para Debian (*packaging helpers*). El más utilizado para el empaquetado en general es `debhelper`, y en el momento de la discusión que nos ocupa se usaban `python-support` y `python-central`, como aparece en el mensaje citado, para empaquetar paquetes relacionados con `python`.

2.1.2 El Comité Técnico.

Por fin, el mensaje comunica una resolución tomada por el Comité Técnico de Debian. La Constitución de Debian (Debian.org, s.f.[i]) establece el Comité Técnico como uno de los órganos del Proyecto capaces de tomar decisiones (los otros serían los Desarrolladores por medio de una votación, el Líder del Proyecto, el Desarrollador individual en su área de trabajo, los Delegados nombrados por el

2013. Sobre la medición de la popularidad de lenguajes de programación, puede consultarse Wikipedia (s.f.[c]).

The Technical Committee may:

1. `[§6.1.1]` Decide on any matter of technical policy.
   This includes the contents of the technical policy manuals, developers' reference materials, example packages and the behaviour of non-experimental package building tools. (In each case the usual maintainer of the relevant software or documentation makes decisions initially, however; see 6.3(5).)
2. `[§6.1.2]` Decide any technical matter where Developers' jurisdictions overlap.
   In cases where Developers need to implement compatible technical policies or stances (for example, if they disagree about the priorities of conflicting packages, or about ownership of a command name, or about which package is responsible for a bug that both maintainers agree is a bug, or about who should be the maintainer for a package) the technical committee may decide the matter.
3. `[§6.1.3]` Make a decision when asked to do so.
   Any person or body may delegate a decision of their own to the Technical Committee, or seek advice from it.
4. `[§6.1.4]` Overrule a Developer (requires a 3:1 majority).
   The Technical Committee may ask a Developer to take a particular technical course of action even if the Developer does not wish to; this requires a 3:1 majority. For example, the Committee may determine that a complaint made by the submitter of a bug is justified and that the submitter's proposed solution should be implemented.
5. `[§6.1.5]` Offer advice.
   The Technical Committee may make formal announcements about its views on any matter. Individual members may of course make informal statements about their views and about the likely views of the committee.

Cuadro 2.2: Algunos poderes del Comité Técnico

Líder del Proyecto para tareas específicas, y el Secretario del Proyecto. La estructura de poder y los procesos de toma de decisiones del Proyecto Debian se tratarán en el capítulo 3, *Formas de gobierno. Doocracia, agencia y apertura*). Véase el cuadro 2.2, *Algunos poderes del Comité Técnico*, para ver cómo la Constitución detalla algunos de sus poderes.

Es decir, el Comité Técnico (Debian.org, s.f.[j]) es el encargado de resolver los problemas «técnicos» dentro del Proyecto Debian. En palabras de uno de sus miembros:[5]

> **dondelelcaro:** So the Technical Committee is basically the appeal board for technical decisions [...] the Technical Committee can decide any technical question. So if the Technical Committee thinks it is a technical question, then we can address it [...] pretty much anything that involves a package in the archive is a technical question, no matter how many social [...] issues it entails.

En principio, sólo puede decidir sobre cuestiones técnicas, como podemos observar en el siguiente correo electrónico:

```
Date: Fri, 11 Sep 2009 15:07:47 -0700
From: Don Armstrong <don@debian.org>
To: debian-project@lists.debian.org
Subject: Re: Distributing software written by hostile upstream developers

On Fri, 11 Sep 2009, Matthew Johnson wrote:
> On Thu Sep 10 12:53, Steve McIntyre wrote:
> > Well, what happens if somebody wants to maintain software where there
> > is a strong set of opinion that we don't want it? In this case, I'd
> > like to delegate the power to the ftpmasters to say so and reject from
> > NEW etc. If we have a clear consensus that that would be OK then fine;
> > otherwise I'd like to run this through the GR process to make sure
the
> > project as a whole agrees.
> > Isn't that a TC job? overruling developers?

The CTTE cannot overrule non-technical decisions of developers. [§6.1.4
only applies to technical decisions.]

I suppose the project leader/ftpmasters could delegate this to the CTTE
under §5.1.4 and the CTTE could make a decision under §6.1.3, but I'm not
sure how that would interoperate with §6.1.4. Presumably the project leader
```

[5] En el momento descrito. En general, lo que se explica aquí sobre el Comité Técnico tiene validez para el momento descrito. Posteriormente se han producido algunos pequeños cambios relativos a su funcionamiento.

```
has the authority to override non-technical decisions that affect the project
under §5.1.4.

Don Armstrong
```

2.1.3 El BTS.

Esta distinción entre cuestiones técnicas y sociales centrará la discusión en la sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*. Volvamos ahora al correo con el que se abría esta sección. En él se dice que se le pidió al Comité Técnico tomar una decisión en #573745, que aparece también en el asunto del correo y al final, donde se da un enlace para obtener más información sobre esta cuestión. #573745 es el número del *bug* en el que se planteó la petición. Un *bug* es un fallo o un defecto en un programa o un sistema. El Proyecto Debian, como muchos otros, mantiene un sistema de seguimiento de los fallos que le afectan, el *Debian Bug Tracking System*, a partir de ahora BTS (véase el cuadro 2.3 *Bug Tracking System*).

Es decir, las cuestiones y problemas sobre las que debe tomar una decisión el Comité Técnico de Debian se tratan formalmente como *bugs* del sistema. Puesto que el Comité Técnico no es un paquete, los informes de fallo se asignan a un *pseudo-paquete*, es decir, un paquete que sólo existe a efectos del sistema de seguimiento de fallos. Un pseudo-paquete no se refiere a un paquete fuente o a un paquete binario. Se usan para poder asignar *bugs* a partes de la distribución o del proyecto que no se originan a partir de un paquete fuente. En este caso el paquete relevante es `tech-ctte` (Bugs.debian.org, s.f.[c]).

Por lo tanto, la resolución de los problemas que el Comité decide que son técnicos se gestiona mediante un proceso que se inserta él mismo en la infraestructura técnica de gestión de los paquetes del Proyecto. Estos problemas –y el que estamos describiendo es quizás uno de los ejemplos más claros– están generalmente mezclados con problemas de comunicación y de interacción entre miembros del Proyecto, problemas que a su vez ellos mismos intentan distinguir continuamente como «sociales» en oposición a los «técnicos». Este aspecto de la práctica de toma decisiones me resultó ciertamente sorprendente, y de hecho fue una de las cuestiones que me hizo detenerme, durante el trabajo de campo, en las cuestiones

El **Bug Tracking System** es el sistema que usa Debian para el seguimiento de fallos. Se puede consultar en Debian.org (s.f.[b]). Se puede encontrar una explicación útil e interesante en Wiki.debian.org (s.f.[e]). Pero esta interface *web* sólo permite consultarla. El envío de informes de fallo (*bug reports*) o la modificación de los existentes se realiza mediante el envío de correo electrónico (normalmente a través de la herramienta de línea de comandos `reportbug`). Cualquiera puede no sólo reportar un *bug*, sino modificarlo mandando un correo. En los correos enviados al BTS es posible y deseable la inclusión de comandos que automáticamente producirán la clasificación del fallo, su asignación a un determinado paquete, su cierre, su reasignación a otro paquete, etc. (por ejemplo, un correo a `control@bugs.debian.org` que incluya la línea `severity 12345 minor` cambiará la severidad del bug #12345 a *minor*). Estos correos contienen información destinada al *software* que lo procesa, pero también al mantenedor humano del paquete implicado:

> Of course, mail to both `control@bugs` and `bugnumber@bugs` contains both commands for the control server and information for humans. In order not to confuse the control parser with human text, there is a special command, **thanks**, that you might have noticed after the control commands. This tell the parser to stop parsing the mail from then on. Please be nice to the parser and use `thanks` to end your control messages. :-) (Wiki.debian.org, s.f.[e])

El BTS reenvía muchos de estos correos, cuando es útil, a los mantenedores del paquete, a determinadas listas de correo, al que ha enviado el informe, o a cualquiera interesado.
De acuerdo con el punto 3 del Contrato Social, «No ocultaremos los problemas» (Véase el apéndice A, *Contrato social*), la lista de fallos es completamente abierta y pública.
Se han reportado más de 700.000 *bugs* a lo largo de la historia del *Debian BTS*.

Cuadro 2.3: *Bug Tracking System*

discutidas en la sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*. Lo que esto muestra es que la distinción entre ambos ámbitos, lejos de estar clara, es un proceso que está siempre abierto a la interpretación y la negociación. Preguntado por las razones del uso del BTS, el mismo miembro del Comité Técnico entrevistado respondía:

> **dondelelcaro:** almost every technical problem involves a change to a package in the archive, so normally what happens is, there is a bug in a package, and when it gets to a point when they can no longer resolve the bug with their problem, it gets elevated to the Technical Committee.

Las ventajas de este procedimiento consisten en que la discusión sobre el problema y las posibles soluciones se integran con la historia del problema en cuestión, que se encuentra ya en el BTS, siendo fácilmente accesibles para los interesados. Y una vez alcanzada una decisión, o bien se cierra el informe de fallo (el *bug*) o, si el Comité decide que no es de su competencia, se reasigna al paquete objeto de la disputa, como es el procedimiento estándar para cualquier *bug*. En cualquier caso, lo que se observa en este procedimiento es la importancia fundamental de los dispositivos técnicos de mediación en el manejo de los problemas suscitados por la interacción entre los miembros del Proyecto.

2.1.4 Resolución del conflicto.

Veamos el informe de fallo del que estamos hablando, tal y como aparece en el BTS. El informe es largo, y contiene algunos detalles que quizás no sean relevantes en este momento, pero creo que merece la pena citarlo en su integridad para comprender el conjunto de prácticas que estamos describiendo:[6]

```
From: Sandro Tosi <morph@debian.org>
To: Debian Bug Tracking System <submit@bugs.debian.org>
Subject: Please decide on Python interpreter packages maintainership
Date: Sat, 13 Mar 2010 17:24:51 +0100

[Message part 1 (text/plain, inline)]
Package: tech-ctte
Severity: normal

Hello Technical Committee,

we'd like you to decide about how the Python interpreter packages should
be maintained in Debian.
```

[6] Se puede consultar, junto con todos los mensajes que le responden, en Bugs.debian.org (s.f.[d]).

The problems that we have with the current situation can mostly be described by having a look at the way the Python 2.6 transition is being handled.

1. The Python maintainer delayed the upload of version 2.6 for 14 months since the first upstream release, without giving a valid technical reason for doing so. At the same time, the same maintainer uploaded said Python 2.6 package quickly to Ubuntu, which released twice with this version as default before it was uploaded to unstable.
2. When finally uploaded, the python2.6 package contained an unplanned transition to a new location for installed modules. This solution was not discussed at all with other maintainers and led to major changes having to be rushed into all packaging tools, and hundreds of packages left to be fixed.
3. The Python maintainer didn’t provide any kind of impact analysis of this transition, leaving all the burden on other maintainers (mostly the Python modules team) and didn’t try to set up dialogue with the team, which was left to file bugs and do NMUs so that packages could build cleanly with these changes. At the same time, the same maintainer provided a very thorough analysis of the upcoming changes in GCC 4.5, collaborating with other developers to test them on a large scale, and thus proving he has all technical and communication skills to do this.
4. The Python maintainer delayed adding python2.6 to the supported versions, asking for python2.4 to be removed first. It is a good thing to remove python2.4 of course, but it could have been done later (or earlier), without delaying the former important step.
5. The Python maintainer didn’t provide an environment where maintainers could test their packages with python2.6 as default in experimental, which is something that was asked since he announced the 2.6 transition. Again, the Python modules team was left the burden of setting up test environments, filing bugs, helping other maintainers to fix their packages, preparing and sponsoring NMUs, etc. When asked about this situation, he replied indirectly (using another developer as a proxy) that he is working on adding additional features unrelated to the transition before this happens.
6. The Python maintainer has still virtually zero communication with the Debian Python modules and Python applications teams, and there seems to have been no significant progress in this.

This situation is not new. Similar problems occurred for previous Python transitions, starting as early as python2.2 and getting worse over the time, despite the increasing amount of work put in Python-related tools and of people involved in Python packaging. A common pattern is that he often blocks important transitions to force some controversial, unrelated technical changes to be implemented before these transitions happen.

```
Given all these points, we think that the current Python maintainer has
no real interest in maintaining Python in Debian, and no interest in working
in an open fashion with other people committed in this area. Therefore,
we.d like to move the maintainership of pythonX.Y* (interpreters and related
packages) and python-defaults (due to its specific role in the Python environment)
packages to a team of people that already showed their involvement in Python
in Debian, their ability to work in a team, and their will to bring a constant
attention to Python in Debian; among them we propose ourselves:

  • Luca Falavigna <dktrkranz@debian.org>
  • Josselin Mouette <joss@debian.org>
  • Sandro Tosi <morph@debian.org>
  • Bernd Zeimetz <bzed@debian.org>
  • anyone else willing to help, including of course the current maintainer,
    provided the above points are met.

Thanks to the amount of work the Python modules team has done for this
transition, Python 2.6 can now become the default version in unstable (of
course after the approval of the Release Team) with minimal breakage, so
that time can be the smoother one to hand the package over to a team.

Thanks for your attention,
Luca, Josselin, Sandro, Bernd

PS: attached you can find the same text along with our signature of it.

– System Information:
Debian Release: squeeze/sid
APT prefers unstable
APT policy: (500, 'unstable'), (1, 'experimental')
Architecture: amd64 (x86_64)
Kernel: Linux 2.6.32-3-amd64 (SMP w/4 CPU cores)
Locale: LANG=en_US.UTF-8, LC_CTYPE=en_US.UTF-8 (charmap=UTF-8)
Shell: /bin/sh linked to /bin/bash

[python_tech-ctte (text/plain, attachment)]
[python_tech-ctte.asc_bzed (application/pgp-signature, attachment)]
[python_tech-ctte.asc_dktrkranz (application/pgp-signature, attachment)]
[python_tech-ctte.asc_joss (application/pgp-signature, attachment)]
[python_tech-ctte.asc_morph (application/pgp-signature, attachment)]
```

El paso de reportar el problema relativo al mantenimiento de `python` que estamos discutiendo como un *bug* al BTS puede ser conceptualizado como una clara operación de traducción en el sentido de la Teoría del Actor-Red, en tanto

que se traduce una cuestión de comunicación en una disputa técnica. Como dice Latour, «las operaciones de traducción transforman las cuestiones políticas en cuestiones técnicas y viceversa. Mientras dura una controversia, las operaciones de persuasión movilizan una mezcla de agentes humanos y no humanos» (Latour, 2001, pág. 120). Y eso es, como vamos a ver, precisamente lo que nos encontramos: una controversia que mezcla agentes humanos y no humanos. De ahí que sea tan importante en esta etnografía describir las mediaciones tecnológicas que se producen. Pero además de este primer sentido más o menos directo de traducción debemos tener en cuenta, como vamos a hacer, el sentido más general de traducción como un proceso de constitución de actores y de colectivos (Callon, 1984).

En los dos años que transcurren entre los dos mensajes citados (el informe de fallo y la resolución del Comité Técnico) no sólo han tenido lugar prácticas de comunicación y discusión entre los actores humanos involucrados, intentos de coordinación más o menos exitosos. El conjunto de acciones sociales que se despliega en esta situación incluye también prácticas que implican la participación de numerosos dispositivos tecnológicos, bien modificando los aspectos técnicos implicados, bien potenciando la capacidad de decisión del Comité Técnico. Veamos una panorámica de cómo se desarrolló y desplegó esta controversia produciendo nuevas prácticas de mediación técnica. De hecho, como veremos en la sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*, el hecho de que lo que en su origen se vivió como un problema de comunicación entre individuos pudiera ser conceptualizado y tratado como un problema técnico depende de la caracterización de los problemas sociales, en oposición a los técnicos, como aquéllos que no admiten (o lo hacen con dificultad) la mediación tecnológica sin controversia.

Tanto en el informe de fallo como en la resolución del Comité Técnico, el problema planteado es fundamentalmente de comunicación y coordinación entre mantenedores de diferentes paquetes, manifestado sobre todo en el retraso en la preparación de la versión 2.6 de `python`, lo que podría impedir su inclusión en la siguiente versión del sistema operativo, y que estaría suponiendo un trabajo extra para los mantenedores de módulos (archivos en lenguaje `python` que implementan una funcionalidad que puede ser usada por diferentes programas) y programas en

python por la falta de coordinación. Tanto módulos como programas dependen para su ejecución de la presencia de una determinada versión del intérprete. Como hemos visto, el conflicto estaría planteado entre el mantenedor de pythonX.Y* (es decir, de las distintas versiones concretas del intérprete python, como python2.6 o python3.1) y python-defaults (el paquete en el que se definen cuestiones como las versiones soportadas y por defecto del intérprete, y que determina la política de empaquetado y distribución), y por otra algunos de los mantenedores de módulos y programas en python. A esto se añade también las dificultades provocadas por la existencia de formas diferentes de preparar los módulos y programas en phyton para su inclusión en Debian, a través de diferentes *packaging helpers*: python-central, mantenido por el mantenedor de python, y python-support, mantenido por uno de los desarrolladores que mandaron el informe de fallo.

Este último problema se fue solucionando por sí solo, al introducirse y generalizarse los *helpers* dh_python2 y dh_python3 (para python3), que dejan obsoletos a los anteriores. Del mismo modo, el paquete python-defaults pasó a sumar dos mantenedores más (uno de ellos el autor de dh_python2), de forma que de hecho parte de las quejas iniciales simplemente dejaron de ser relevantes. Al haber cambiado la morfología de los actores por su diferente composición, cambian sus formas de interacción. También se puede observar en las discusiones al respecto una opinión generalizada de que ha mejorado notablemente la discusión y la coordinación de la política de empaquetado en el seno de los que mantienen el *ecosistema* de python en Debian, en su lista de correo y el canal de IRC.

Incidentalmente, podemos señalar que no es raro que los problemas presentados al Comité Técnico se solucionen antes de que éste adopte una resolución. Como dice un miembro del mismo:

> **vorlon:** I am a member of the Technical Committee. It is a [...] party of gray hairs, gray beards, that is very slow to take decisions. So frequently what happens is, someone in the heat of the argument with a maintainer, will ask [...] something to the Technical Committee, and the TC will deliberate for 2 or 3 months at the time, and fail to reach a decision. And by the time they actually try to reach a decision, the

> people that were having [a dispute] have settled it down on their own. The problem goes away. I can't say I am proud that the TC works so slowly, but in the end things work out for the most part anyway.

Acabo de decir que algunos problemas se solucionaron «por sí solos». Y sin embargo, estrictamente, nada más lejos de la realidad. Se solucionaron gracias a un conjunto de mediaciones técnicas, de prácticas en relación a los paquetes implicados. Sólo podríamos afirmar que se solucionaron por sí solos si no tenemos en cuenta la relación entre los problemas de comunicación, coordinación y relación por un lado, y estos procesos de mediación técnica por el otro. Es decir, si damos por supuesto que los problemas de comunicación sólo se resuelven con más comunicación entre los sujetos implicados. Y sin embargo, como señaló repetidamente uno de los miembros del Comité Técnico a lo largo de la discusión, entre las obligaciones formales de un mantenedor en Debian no está la de comunicarse pública o privadamente ni siquiera con aquéllos interesados en su trabajo. El problema con la diversidad de *helpers* no se resolvió discutiendo cómo habían de ser, sino *escribiendo* uno nuevo. La práctica de los nuevos mantenedores del paquete *python-defaults* consistió más en *implementar* cambios en la política de empaquetado de python que en convencer de sus bondades.[7] Ambas cuestiones no son independientes, puesto que los *helpers* se incluyen dentro de los paquetes `python-defaults` y `python3-defaults`. En un mensaje del 20 de marzo de 2012, uno de sus mantenedores escribe:

> I would propose that the tech ctte not consider changing maintainership of python-defaults and python3-defaults. They are both maintained by a team of three DDs and seem to be working reasonably well. A number of changes for the better have been made in the last two years and these packages are essential to them.
>
> There is agreement on a single helper package for Python 3 (dh_python3). It is implemented in python3-defaults. There is general (not 100%, but general) consensus on gradual migration to a single helper for Python

[7] Esto no quiere decir que esas escrituras e implementaciones no tengan a su vez una dimensión retórica. Véase la sección 4.3, *Instrumentalidad y expresividad del código*.

> (dh_python2). Both python-support and python-central are now classified as deprecated with the agreement of their maintainers. dh_python2 is part of python-defaults.
>
> As I read the tech ctte discussion, the goal of having reasonable team maintenance for python-defaults and python3-defaults has been met and further changes are not appropriate.

Lo que se presentaba en principio ante un profano como un problema de comunicación, es experimentado por los participantes como un problema técnico. O cuando menos, la posibilidad y las dificultades de su solución se hacen pasar por su reconducción a una discusión técnica que implica cambios en el funcionamiento de determinados paquetes de *software*. Un año antes, un miembro del Comité Técnico escribía lo siguiente (Debian-ctte, 2011a):

> - if everyone had been constructively engaging on this problem to begin with, the occasion for the current python maintainer to engage in brinksmanship with the python modules team would never have arisen;
> - solving the python helper Problem (which appears to be happening) removes the primary technical point of friction between Matthias and the modules team;
> - failing to solve the policy for python helpers would mean there are outstanding **technical** disagreements that need to be addressed, not just social ones;

La última afirmación resulta llamativa. No conseguir ponerse de acuerdo mostraría no tanto que existen problemas de comunicación y de relación social, cuanto desacuerdos «técnicos» que se contraponen a los «sociales». La condición de posibilidad de resolución de un problema pasa entonces por la posibilidad de su traducción a una discusión técnica. Y sin embargo, como veremos en la sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*, esto es algo que está permanentemente sujeto a negociación y reelaboración. Podemos

ver aquí la tensión existente con la primera afirmación, según la cual el origen del problema es una cuestión de actitud personal por parte de los implicados. Esta tensión se manifiesta continuamente a lo largo de la discusión. Pero lo que esto nos muestra no es una contradicción o una inconsecuencia, sino el proceso abierto mediante el que estos agentes dan forma a su acción social, el proceso por el que crean, recrean y evalúan las reglas que rigen sus prácticas y se expresan en ellas.

2.1.5 Dispositivos que potencian la agencia.

Antes de pasar a la siguiente sección, mencionaré brevemente tres conjuntos más de prácticas, relacionadas entre sí, que afectan más al funcionamiento del Comité Técnico que a la específica discusión que estamos tratando, potenciando su capacidad de decisión. Juntas, nos permiten observar las tensiones entre lo que debe permanecer privado y lo que debe ser público, entre la apertura deseada y la necesidad ocasional de privacidad. El Proyecto Debian se caracteriza por la apertura de sus procesos, como vimos en la introducción y se establece en su Contrato Social. Dos de esas prácticas suponen, en el periodo que estamos considerando, un progresivo esfuerzo por parte del Comité Técnico para ser más abierto y transparente: la celebración de reuniones periódicas en IRC y la publicación de las mismas, y la creación de un repositorio público en el que quede reflejado todo lo relativo a las cuestiones discutidas por el Comité Técnico. Son dos prácticas caracterizadas por la implementación de una mediación tecnológica que cambia la forma de la relación social. Pero también nos encontramos con la tercera, la solicitud para poder tener explícitamente discusiones en privado, entre sí y con los implicados.

2.1.5.1 El robot *Meetbot*.

En abril de 2012, el *Debian Project Leader* o DPL (véase la sección 3.2, *Toma de decisiones: de la meritocracia a la doocracia*) propuso al Comité Técnico la realización de reuniones periódicas por IRC, que garantizaran una revisión periódica de las cuestiones abiertas pendientes. Estas reuniones además harían más transparente el trabajo del Comité Técnico. Así, en un encuentro presencial producido en la conferencia anual de Debian de 2012 con el Comité Técnico,[8] sus miembros animaban a todos los interesados a entrar como espectadores en dichas

[8] Se puede ver el video de la misma en Meetings-archive.debian.net (s.f.[a]).

reuniones. Y no sólo pueden entrar en el momento de la reunión, puesto que los *logs* de las mismas se archivan y pueden ser consultados posteriormente (Meetbot.debian.net, s.f.). Además de la transcripción íntegra de la reunión, se puede consultar también el resumen, mucho más breve, producido por el *Bot* o robot MeetBot (Wiki.debian.org, s.f.[f]). Éste es un programa que va recogiendo, automáticamente y a instancias de los participantes, información como los puntos tratados, los acuerdos tomados, o los participantes activos. En el cuadro 2.4, *Log de MeetBot*, vemos uno de los *logs* de MeetBot tal y como se ve en un navegador.

Corresponde a la sesión que tuvo lugar el 26 de julio de 2012. Podemos ver resumida la información más relevante que se trató, así como los participantes, y los acuerdos tomados. También aparecen numerosos enlaces a través de los que podemos acceder al *log* completo de la reunión, o a las partes específicas de una discusión que nos interese.

2.1.5.2 El repositorio `git` del Comité Técnico.

En segundo lugar, y poco después (Debian-ctte, 2012a), un miembro del Comité Técnico anunció la creación de un repositorio de `git` (un Sistema de Control de Versiones Distribuido; véase el Cuadro *2.5*, Sistemas de Control de Versiones y Repositorios; se ampliará la explicación sobre estos sistemas en la sección 4.8, *Repositorios de* software *libre*) para mantener en un mismo sitio los documentos relativos a su actividad. En el cuadro anterior podemos ver también que se anunció igualmente en la reunión a través de IRC.

Este repositorio es públicamente accesible (Salsa.debian.org, s.f.[b]). Es un conjunto de archivos que podemos bajar, y que contiene la documentación relativa a las cuestiones pendientes, las resueltas, los *logs* de las reuniones mencionadas, o la agenda del Comité, documentos todos a disposición de quien quiera consultarlos. Pero además, al estar mantenido por un Sistema de Control de Versiones, es posible acceder también a la historia de todos los cambios producidos: quién ha cambiado qué, cuándo, o por qué razón.

Juntas, estas dos novedades en la forma de actuación del Comité Técnico suponen un aumento significativo de la apertura de sus procesos y acciones. No como una cuestión de principio o declaración de intenciones, puesto que ya antes

Meeting summary

1. **Who is here?** (dondelelcaro, 17:02:30)
2. **Git repository git://git.debian.org/git/collab-maint/debian-ctte.git** (dondelelcaro, 17:04:11)
3. **Next Meeting Time?** (dondelelcaro, 17:05:27)
4. **#573745: python maintainer [vorlon to write up resolution]** (dondelelcaro, 17:07:59)
5. **#636783: super-majority conflict;** (dondelelcaro, 17:09:34)
    a. **AGREED**: postpone #636783 until open issues are resolved (dondelelcaro, 17:11:48)
6. **#681419 Depends: foo | foo-nonfree** (dondelelcaro, 17:12:04)
    a. **AGREED**: flesh out #681419 Depends: foo | foo-nonfree options via e-mail (dondelelcaro, 17:24:12)
7. **#681687 missing mime entry** (dondelelcaro, 17:24:21)
8. **#681783 Recommends** (dondelelcaro, 17:33:04)
9. **#681783 Recommends in meta packages** (dondelelcaro, 17:33:53)
    a. **ACTION**: Diziet to write up statement to dispose of #681783 with possible reassignment to policy (dondelelcaro, 17:41:49)
10. **#682010 mumble/celt client/server compatibility** (dondelelcaro, 17:42:15)
    a. **ACTION**: dondelelcaro request Ron follow up with upstream about speex interop within two weeks (dondelelcaro, 18:13:31)
11. **TC mailing list / BTS usage [http://lists.debian.org/msgid-search/20494.51776.384105.199244@chiark.greenend.org.uk]** (dondelelcaro, 18:13:5
    a. **ACTION**: Diziet to look into list mail-follow-up-to etc. configuration (Diziet, 18:16:32)
12. **additional business** (dondelelcaro, 18:17:02)

Meeting ended at 18:24:55 UTC (full logs).

Action items

1. Diziet to write up statement to dispose of #681783 with possible reassignment to policy
2. dondelelcaro request Ron follow up with upstream about speex interop within two weeks
3. Diziet to look into list mail-follow-up-to etc. configuration

Action items, by person

1. Diziet
    1. Diziet to write up statement to dispose of #681783 with possible reassignment to policy
    2. Diziet to look into list mail-follow-up-to etc. configuration
2. dondelelcaro
    1. dondelelcaro request Ron follow up with upstream about speex interop within two weeks

People present (lines said)

1. rra (149)
2. Diziet (122)
3. dondelelcaro (110)
4. vorlon (72)
5. ChrisKnadle (7)
6. MeetBot (5)
7. Mithrandir (4)
8. Maulkin (1)

Cuadro 2.4: Log de MeetBot

Un **Sistema de Control de Versiones** es un sistema que guarda los cambios en un archivo o grupo de archivos, normalmente de código fuente, a través de versiones incrementales de los mismos a través del tiempo. Se puede acceder a los cambios en cada una de las versiones, y recuperarlas.

> At its core is a repository, which is a central store of data. The repository stores information in the form of a filesystem tree –a typical hierarchy of files and directories. Any number of clients connect to the repository, and then read or write to these files. By writing data, a client makes the information available to others; by reading data, the client receives information from others. [...]
>
> What makes the [...] repository special is that it remembers every change ever written to it –every change to every file [...]
>
> When a client reads data from the repository, it normally sees only the latest version of the filesystem tree. But the client also has the ability to view previous states of the filesystem. For example, a client can ask historical questions such as "What did this directory contain last Wednesday?" and "Who was the last person to change this file, and what changes did he make?" These are the sorts of questions that are at the heart of any version control system: systems that are designed to track changes to data over time (Collins-Sussman *et al.* 2008, pág. 1).

Cuadro 2.5: Sistemas de Control de Versiones y Repositorios

los documentos fundamentales de Debian, la Constitución y el Contrato Social, la establecían, sino porque esa apertura se inscribe, se delega efectivamente en dispositivos técnicos que la potencian y le dan un mayor alcance. Estos dispositivos técnicos de inscripción contribuyen significativamente a la constitución de formas de acción y de relación social, en este caso construyendo su apertura y transparencia. Sus consecuencias son también importantes para, por ejemplo, un etnógrafo que no se interese sólo por las decisiones tomadas y los resultados de la acción, sino también y especialmente por los procesos que los han constituido:

> we need to have infinite respect for the deciphering of *inscriptions*.

> To propose the description of a technological mechanism is to extract from it precisely the *script* that the engineers had transcribed in the mechanisms and the automatisms of humans or nonhumans. It is to retrace the path of incarnation in the other direction. It is to rewrite in words and arguments what has become, what might have become, thanks to the intermediary of mechanisms, a mute function (Latour, 1996, pág. 206).

Según Latour, el de inscripción «es un término general que hace referencia a todo tipo de transformaciones, es decir, transformaciones a través de las cuales una entidad se materializa en un signo, en un archivo, en un documento, en un trozo de papel, en una huella» (Latour, 2001, pág. 365). Veremos algo más sobre la inscripción de programas de acción en la sección 2.2.4, *Un ejemplo: el* Firmware *y la inscripción de programas de acción*.

2.1.5.3 Discusiones en privado.

La tercera y última cuestión se refería a la posibilidad, para el Comité Técnico, de tener discusiones en privado. Al respecto se puede consultar el mensaje que inicia la discusión (Debian-ctte, 2012b) y el hilo de discusión que le sigue. En ese mensaje se plantea una reforma de la Constitución que lo permita. Y es que el punto 6.3.3 de la Constitución de Debian establece:

> `[3]` Public discussion and decision-making.
>
> Discussion, draft resolutions and amendments, and votes by members of the committee, are made public on the Technical Committee public discussion list. There is no separate secretary for the Committee.

En la discusión sobre el mantenimiento de `python` que estamos desarrollando, esta cuestión surgió repetidamente. Sobre todo ante la no participación pública del mantenedor en las discusiones, diferentes miembros del Comité Técnico discutieron el asunto en privado con él y con otros. A su vez, esto provocó quejas de otros implicados, pero también un reconocimiento general de que era un elemento necesario como forma de mediación personal en la comunicación. Otras

situaciones tratadas por el Comité Técnico han seguido un patrón similar. En el hilo de discusión que se acaba de citar se reconoce ampliamente que es una necesidad del papel de mediación, que en ocasiones ha de asumir el Comité Técnico, el poder tratar cosas en privado. Pero puesto que va en contra de la apertura y la transparencia que según todos ha de caracterizar las acciones dentro del Proyecto, debe ser mantenido en un nivel mínimo. Éste es un primer elemento de tensión con los dos conjuntos de acciones discutidos anteriormente y que tenían como efecto precisamente un aumento de la apertura. Pero hay otro elemento de tensión que resulta muy significativo: tanto la realización de reuniones periódicas por IRC como el mantenimiento de un repositorio del Comité Técnico aumentaban la transparencia del mismo gracias a su mediación por un dispositivo técnico. Ahora la situación es diferente. En ese hilo escribía el Debian Project Leader de ese momento (Debian-ctte, 2012c):

> But the proposed solution seems flawed in at least two ways. The first one is that once easy mechanisms for discussing privately are in place (e.g. a debian-ctte-private list), it will come very naturally to discuss privately also matters that could've been discussed publicly. The second one is that nobody outside the tech-ctte will be able to control what is actually being discussed on those fora.

En este caso, la mediación técnica que facilitaría el mantener discusiones en privado sería muy fácil de implementar técnicamente, pero se percibe como una mediación que trabajaría en contra de los fines y principios que regulan el Proyecto. Y es que, como estamos intentando mostrar, los dispositivos técnicos tienen consecuencias en la forma de las acciones y las relaciones sociales, y constituyen la comunidad de una determinada manera.

Pero hay más. La necesidad de los intercambios privados se vincula a aquellas cuestiones que son caracterizadas como «sociales» (en oposición a las «técnicas»), o con un fuerte componente social. En el último mensaje citado se escribía también:

> Transparency is hard. But it is a worthwhile goal. Out of hypocrisy, I

> suspect that today every non trivial (social) conflict solving will end up having some private exchanges. Especially when mediations are needed. The question we should ask ourselves, then, is how to **minimize** those exchanges. Ideally trying to reduce them to 0 even if we know we're not there yet.

Y en el mismo hilo podemos leer lo siguiente en un mensaje de un miembro del Comité Técnico (Debian-ctte, 2012d):

> Please don't confuse this with the separate question of whether the tech-ctte should get more involved in social issues. This is an independent question. The ability to raise an initial question in private is valuable for technical conflicts with a social component, where the person raising the technical conflict is concerned that just opening a bug against tech-ctte will come across as immediate escalation. It can be very useful to get a sanity check of one's reasoning before taking that step, as well as assistance in how to word a proposal to be less confrontational, and that's one of the places where I think this option will be used.

Nos podríamos ver tentados de establecer una correlación entre cuestiones técnicas, mediación técnica y transparencia por un lado, y cuestiones sociales, falta de mediación y privacidad por el otro. Entre la mediación técnica y la transparencia. Sin embargo, también hemos visto como la privacidad podría ser implementada técnicamente con facilidad. Lo que creo que encontramos aquí es una manera de construir precisamente la diferencia entre lo que es técnico y lo que es social. Se construyen como sociales aquellas cuestiones o problemas que se resisten a cerrarse implementando ese tipo específico de mediación, la tecnológica. Lo veremos en la sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*.

Nos hemos encontrado aquí con dos sentidos del término «mediación» aparentemente diferentes. Por un lado, la mediación técnica entendida como la ins-

cripción, la implementación en un dispositivo técnico de una determinada solución (o programa de acción; véase especialmente la sección 2.2.4, *Un ejemplo: el* Firmware *y la inscripción de programas de acción*). Por otro lado, la mediación como la intervención de un tercer sujeto en la disputa entre otros dos para mitigar sus problemas de comunicación. Sin embargo, ambos sentidos comparten lo esencial, que la mediación no es sino una manera de traducir, y por lo tanto transformar y desplazar, los fines de los agentes implicados. Sobre los sentidos de los conceptos de mediación y mediador véanse la sección 4.2.2, *Mediadores e intermediarios. Las figuras del* bug *y del* hacker, y Latour (2001, cap. 6).

2.1.6 Agentes, humanos y no humanos.

En esta sección hemos empezado a comprender cómo estos agentes dan forma a su relación social, y cómo su acción se dirige también a crear y transformar las reglas que le dan forma. Para eso es fundamental entender cómo interactúan con dispositivos tecnológicos, y cómo las acciones de los miembros de Debian y de estos dispositivos se constituyen mutuamente. La práctica técnica de un mantenedor con un paquete (en el caso considerado, `python` y paquetes relacionados) crea determinados vínculos y condiciona las prácticas de otros, construyendo el colectivo de una determinada manera.

¿Qué agentes han ido apareciendo en esta descripción? En primer lugar, el mantenedor de `python`, los que presentaron el informe de fallo, el Comité Técnico, el Debian Project Leader, y demás personas interesadas en el desarrollo y mantenimiento de `python` en Debian. Pero también el BTS, las listas de correo, los paquetes de *software*, la Constitución de Debian o los repositorios. Lo interesante aquí no es tanto que estos dispositivos tengan o no agencia, sino mostrar que ésta es distribuida, que los actantes son tanto *sub* como *supra*individuales.[9] Los agentes lo son en tanto que se constituyen como composición y articulación de estos actantes. *Todos* los agentes mencionados en este párrafo tienen agencia en tanto que a través de ellos actúan también una gran cantidad de actantes, que los constituyen y a los que constituyen.

Ofrecer una descripción etnográficamente adecuada de un proceso como el

[9] Véase por ejemplo Latour (2001, 214 ss.).

expuesto aquí requiere atender a cómo se reparte y comparte la agencia entre los diferentes actores implicados. Porque los agentes sólo se constituyen como tales a partir de las relaciones que establecen entre ellos, son un producto del conjunto de acción tanto como su condición de posibilidad. Hemos asistido a un proceso en el que se han constituido nuevas relaciones sociales, y con ellas, nuevos agentes. El mantenedor de un paquete sólo se constituye como tal a partir de la relación que establece con un determinado *software*: los nuevos mantenedores de `python-defaults` aparecen como agentes relevantes cuando introducen su nombre como tales mantenedores en uno de los archivos que constituyen el paquete. El Comité Técnico se convierte en un agente diferente, en un cuerpo diferente, cuando usa el BTS, o un repositorio distribuido, o cuando discute en sesiones de IRC a las que cualquiera puede asistir. Se es un agente diferente al reportar un informe de fallo en el BTS que al escribir un correo privado a un miembro del Comité Técnico. En definitiva, a lo largo de todo este proceso vemos surgir una comunidad o colectivo «python en Debian» diferente a la que existía previamente, y a la que se puede implicar en las discusiones (fundamentalmente a través de su lista de correo).[10]

Y en la configuración de esta comunidad o colectivo es esencial el papel configurador de la acción que juegan los dispositivos técnicos. Estos contribuyen, y mucho, a la configuración de los flujos de acción social y de las cadenas de prácticas. Sobre todo, a la configuración de los actores mismos. Imaginemos una situación en la que se acaba de crear el Comité Técnico, y se le pide que resuelva sobre el problema que hemos considerado. Pero no utiliza el BTS y no tiene asignado un pseudo-paquete ni dispone de una lista de correo, ni de ninguno de los elementos que hemos descrito. Sin duda, el resultado final hubiera sido muy diferente. Como poco, con toda probabilidad el asunto se hubiera disuelto en los dos años que duró la discusión sin la consistencia proporcionada por esos elementos. El conjunto de acción y los agentes que lo producen y son producidos por él habrían sido muy diferentes. Lo mismo hubiese ocurrido si no fuese posible la existencia de más de un *packaging helper*, por ejemplo.

[10] También han cambiado las relaciones entre Debian como distribución y el *upstream* (los desarrolladores originales) de `python`, así como con Ubuntu, una distribución derivada. Sobre los problemas de la relación de Debian con diferentes *upstreams* y *downstreams* (como Ubuntu), véase la sección 4.6.1, Debian como *Proxy*. *Upstream* y *downstream*.

¿Basta esto para incluir esos elementos en la lista de agentes, incluso aunque no tengan conciencia ni intencionalidad? Como dice Díaz de Rada, «si una máquina pudiera por su cuenta crear reglas convencionales y reglas para juzgar esas reglas, entonces deberíamos reconocer que esa máquina es agente de cultura» (Díaz de Rada, 2010, pág. 239). Si nos atenemos estrictamente a la cita, esos dispositivos técnicos no serían agentes de cultura, no tendrían agencia cultural, en tanto no crean reglas convencionales «por su cuenta». Pero es que entonces tampoco lo serían los humanos que los usan, porque tampoco lo hacen «por su cuenta». Es en la interacción entre ambos donde se crean las convenciones. No es sólo que los humanos crean convenciones con las limitaciones y posibilidades que les ofrecen los dispositivos. Es al usarlos cuando surgen esas convenciones, por acción de unos y de otros.

2.1.7 Procesos de traducción.

En definitiva, lo que hemos estado viendo son prácticas que se engloban en lo que Callon llama «traducción»: «the notion of translation emphasizes the continuity of the displacements and transformations which occur in this story: displacements of goals and interests, and also, displacements of devices, human beings, larvae and inscriptions» (Callon, 1984, pág. 223).[11] Encontramos una definición similar en Latour:

> Por sus connotaciones lingüísticas y materiales, la palabra traducción se refiere a todos los desplazamientos que se verifican a través de actores cuya mediación es indispensable para que ocurra cualquier acción. En vez de una oposición rígida entre el contexto y el contenido, las cadenas de traducción se refieren al trabajo mediante el que los actores modifican, desplazan y trasladan sus distintos y contrapuestos intereses (Latour, 2001, pág. 370).

De hecho, no sería difícil la reconstrucción del proceso presentado en términos de los cuatro momentos de la traducción que propone Callon. Así, encontramos un momento de *problematización*, cuando los proponentes del *bug* realizan

[11] Sólo habría que cambiar las *larvae* por los no tan diferentes *bugs*.

el doble movimiento (Callon, 1984, 203 ss.) de definir a los actores implicados y sus intereses: el mantenedor de `python` y su interés centrado en Ubuntu, el resto de mantenedores de paquetes relacionados y los problemas extra que tienen, el *python modules team* y su labor resolviendo los problemas planteados; y de situarse ellos mismos, su interpretación de la situación, como «punto de paso obligatorio» (Callon, 1984, pág. 205): «Therefore, we.d like to move the maintainership of pythonX.Y* (interpreters and related packages) and python-defaults (due to its specific role in the Python environment) packages to a team of people that already showed their involvement in Python in Debian, their ability to work in a team, and their will to bring a constant attention to Python in Debian; among them we propose ourselves», como puede verse claramente en el informe de fallo. El establecimiento de un punto de paso obligatorio implica que los demás actores acepten la definición que se ha hecho de la situación, y que la solución se juega en ese terreno. Que los intereses de todos los actores converjan en ese punto (Callon, 1984, pág. 205). En este caso, que lo fundamental de toda la situación es decidir quién debe ser el mantenedor de `python`.

Observamos también un momento de *interesamiento*, «the group of actions by which an entity [...] attempts to impose and stabilize the identity of the other actors it defines through its problematization. Different devices are used to implement these actions» (Callon, 1984, págs. 207-208). Todo el proceso de discusión sobre los intereses de los actores implicados y su interpretación caería bajo esta categoría. En este caso, este proceso de interesamiento es muy disputado, con continuas discusiones sobre los intereses y el papel que han de jugar las diferentes partes (los mantenedores, el Comité Técnico, *upstream*, o las distribuciones derivadas). Pero también incluye el desarrollo de dispositivos: «To interest other actors is to build devices which can be placed between them and all other entities who want to define their identities themselves» (Callon, 1984, pág. 208). El ejemplo más claro de construcción de dispositivos que encontramos aquí es el desarrollo de los nuevos *helpers* para `python` y de la *policy* implementada en `python-defaults`, que cambian efectivamente las relaciones entre las diferentes partes eliminando la fragmentación producida por los diferentes flujos de trabajo. Que estos dispositivos no sean obra del mantenedor de `python`, ni de quienes propusieron el cambio, ni del Comité Técnico, muestra que (mucho más incluso que en el caso analizado por Callon), el proceso considerado es abierto y la agencia

en él enormemente repartida.

Otros dispositivos que se intentan construir para separar a los actores de otras entidades que pretenden definir su identidad incluyen, por ejemplo, el intento de impedir (o mantener en el mínimo) las conversaciones privadas, o el intento de cambiar las relaciones con *upstream* y con la distribución derivada Ubuntu (véase la 4.6.1, *Debian como* Proxy. Upstream *y* downstream). Una parte de la complejidad del mantenimiento de `python` en Debian proviene de las desviaciones[12] en la instalación respecto de cómo se haría a partir de *upstream* directamente. Así que una propuesta que recibió un acuerdo generalizado (aunque finalmente no se realizó) fue la de integrar como mantenedor a una persona que trabajaba en el *upstream* de `python`. Más complejas son las relaciones con Ubuntu. En este caso en particular, los firmantes del informe de fallo interpretaban que una parte del problema con el mantenimiento de `python` en Debian era la mayor preocupación de su mantenedor por mantenerlo propiamente en Ubuntu que en Debian, lo que llevaba a soluciones inadecuadas o tardías para Debian. De hecho se llegó a proponer que, de admitir nuevos mantenedores para `python`, estos no estuvieran además en Ubuntu. En los términos propuestos por Callon, se trataba de impedir la definición de los mantenedores de `python` por parte de otro actor, en este caso Ubuntu. Un destacado miembro del Comité Técnico, también Desarrollador de Ubuntu, contestó repetidamente ese intento, que finalmente no prosperó.

En tercer lugar, el *enrolamiento*

> designates the device by which a set of interrelated roles is defined and attributed to actors who accept them. Interessement achieves enrolment if it is successful. To describe enrolment is thus to describe the group of multilateral negotiations, trials of strength and tricks that accompany the interessements and enable them to succeed (Callon, 1984, pág. 211).

Esto es justamente lo que he estado describiendo en esta sección. Y si de hecho los que firmaron el informe de fallo no consiguieron su objetivo inmediato,

[12] Sobre todo respecto a donde se instalan los paquetes. Véase Wiki.debian.org (s.f.[d]).

cambiar el mantenedor de `python`, sí que a lo largo del proceso se ha producido la creación y reacomodo de las relaciones entre los actores implicados, al tiempo que han surgido otras relaciones y otros actores nuevos. Se configura así una forma de vida y se transforma una serie de cuestiones iniciales en proposiciones aceptadas como realidad (Callon, 1984): la *policy* de `python` está mejor definida, el mantenimiento de `python` se realiza más en equipo, existe acuerdo sobre los *packaging helpers*, la relación con *upstream* y Ubuntu es más fluida, existe una «comunidad `python` en Debian» dinámica.

Por último, la *movilización* de aliados. Para Callon, este momento consiste en la constitución de representantes y portavoces de los agentes implicados (Callon, 1984, 214 ss.). O más exactamente, en la constitución de esos grupos a través de sus portavoces. En el informe de fallo se observa cómo sus firmantes pretenden ser portavoces (en el sentido de que dan una interpretación, no de que afirmen haber obtenido una representación legítima y expresa) de quienes empaquetan módulos y programas de `python`, así como de las necesidades y objetivos del Proyecto. También, a lo largo de toda la discusión, al presentar evidencias de los problemas que la situación produce en la integración de los paquetes en la distribución, constituyen el conjunto de paquetes relacionados con `python` de una determinada manera. Y ya hemos visto cómo otros agentes implicados ofrecen construcciones diferentes de estas realidades. En toda esta controversia los diferentes agentes van construyendo una nueva realidad movilizando (y *traduciendo*) lo que pretenden que sean aliados: miembros del Comité Técnico, otros mantenedores, pero también los paquetes, los *helpers*, la *policy*, o informes de fallo en paquetes relacionados con `python`.

En definitiva, a lo que asistimos es a un proceso que contribuye a la reconstitución de actores y colectivos. Contribuye, porque el presentado aquí es tan sólo una parte pequeña de los que conforman el Proyecto, pero nos ha permitido ver algunos mecanismos generales de manera más concreta. Se puede afirmar lo mismo que Callon respecto a la domesticación de vieiras en la bahía de St. Brieuc:

> The initial problematization defined a series of negotiable hypotheses on identity, relationships and goals of the different actors. Now at the

> end of the four moments described, a constrained network of relationships has been built (Callon, 1984, pág. 218).

En su práctica social, todos los participantes en esta discusión ponen en relación sus objetivos con elementos tecnológicos, formas de comunicación, actitudes de otros participantes, o los objetivos y principios del Proyecto. Y proponen nuevas formas de asociación entre estos elementos, creando una nueva realidad reflejada en diferentes proposiciones: «`python` debe ser mantenido colaborativamente», «las conversaciones privadas en el comité deben mantenerse en un mínimo», o «la situación de `python` en Debian ha mejorado notablemente» son algunos ejemplos.

Esta realidad colectiva es el producto de toda una series de cadenas de prácticas (que hemos empezado a seguir, y cuyo despliegue continuará a lo largo del resto de este trabajo) que son cadenas de operaciones de traducción y de mediación. Mediación tecnológica y mediación personal (un término también muy empleado en el Proyecto Debian sobre todo en la resolución de conflictos), también mediación de las reglas establecidas en la Constitución o en las diversas *policies* y de la relación con el entorno del *software libre* (*upstream*, distribuciones derivadas y usuarios principalmente). De ahí la importancia de la cita que abre este capítulo: lo social aparece en la práctica de los actores entendidos como cadenas de mediadores.

2.2 Primera controversia: Sobre la diferencia entre lo técnico y lo social.

En la sección 2.1, *Bug #573745*, ha empezado a aparecer una distinción que se ha ido revelando como fundamental a lo largo del trabajo de campo: la distinción entre los ámbitos «técnico» y «social» dentro del Proyecto Debian. Intentaré mostrar ahora cómo se construye esa distinción a partir de las prácticas y los discursos de los integrantes del Proyecto. Una vez realizado lo anterior, lo pondré en relación con una teoría de la agencia que dé cuenta de cómo se hace inteligible precisamente como proceso de redistribución de la agencia a lo largo del entramado sociotécnico que estamos considerando.

La distinción mencionada es una distinción *emic* que utilizan los miembros de Debian, aunque no siempre del mismo modo. En esta distinción, el término dominante es «técnico», en tanto que puede obtenerse una definición o especificación del mismo más clara. Lo «social» suele quedar como una categoría residual que recoge aquello que no es claramente «técnico». Así, en las entrevistas es mucho más fácil obtener una especificación de la dimensión técnica del Proyecto que de la social. Veamos algunos ejemplos de cómo se expresa esta diferencia en general. Luego veremos cómo se produce en el contexto del caso discutido en la sección anterior.

La primera vez que me encontré con esta distinción fue en una entrevista con un Desarrollador de Debian, que mencionó que en Debian, en ocasiones, «se dan soluciones técnicas a problemas sociales». Al pedirle ejemplos de ese tipo de situaciones, indicó los cambios en los sistemas de votación, o las propuestas de reforma del proceso de pertenencia al Proyecto Debian que implican la distinción entre diferentes estatus de miembros con diferentes privilegios. Al continuar con el trabajo de campo, se hizo manifiesto que esta expresión funciona como un lugar común en las discusiones en el seno del Proyecto, y apareció repetidamente en diferentes entrevistas realizadas, así como en mensajes a listas de correo y *blogs* personales.

En esta expresión subyace además un sentido de autocrítica, en tanto que se utiliza para advertir de un peligro que se considera propio de la interacción entre personas dedicadas a una actividad tan técnica como la que nos ocupa. Normalmente, «dar una solución técnica a un problema social» se considera dar una solución equivocada:

> Funny how technical people always seem to search for technical solutions to social problems (Verhelst, 2008).

> It has been said that it is a common folly of a technophile to attempt to employ technical solutions toward solving social problems (Hill, 2005).

> Debian's freeze sounds like a technical hack to address a social pro-

> blem, and that disturbs me a bit (Nussbaum, 2008a).

> Our core teams not communicating is a social problem. RT is a technical solution. There are no technical solutions for social problems (Debian-project, 2007a).

O como lo expresa un Desarrollador con mucha claridad, indicando además la dificultad de eliminar este tipo de situaciones:

> One of these [mistakes] is trying to fix some social issue with a technical measure. Unfortunately, given the technical orientation of most of the developers, this appears from time to time in our Debian private mailing list, and yesterday I realized it's our own version of Godwin's law:
>
> "As a social problem discussion grows longer in debian-private the probability of some developer proposing a technical solution aproaches one."
>
> Not discussing about this problems in debian-private would be a good start, but of course that would only change the name of the list in the above sentence ;-) (Mones, 2014).

Como etnógrafos, debemos extrañarnos aquí ante lo obvio. La diferencia entre lo técnico y lo social aparece como obvia para los miembros de Debian (el hecho de que existe esa distinción, por más que existan casos problemáticos, como veremos), que la usan constantemente. Y también para el antropólogo, que en principio se interesaría por lo propiamente social y podría dejar de lado la preocupación por lo técnico. Pero es necesario extrañarse ante la proliferación explícita de esta distinción. Vamos a ello.

2.2.1 Concepciones *emic* de lo técnico y lo social.

Así que a lo largo del trabajo de campo fui preguntando a los miembros de Debian por cómo debía entenderse esta distinción. Algunos fragmentos (algo ex-

tensos, pero muy ilustrativos) de entrevistas servirán para ilustrar las diferentes concepciones de estos términos y sus diferencias:

> **dato:** Para mí un problema técnico es todos esos en los que somos muy competentes y estamos en nuestra salsa y es muy fácil resolver. Bueno a veces no, pero tienen argumentaciones meramente de soluciones técnicas detrás y que no tienen más complicaciones, es lo que somos buenos haciendo. Pues eso, que si hay un *bug* que no sabemos resolver, pues simplemente hablamos de las soluciones, y decimos cuál es la mejor, o el encargado decide cuál es la mejor, y seguimos adelante. Y luego están todos los demás, que surgen inevitablemente, que no tenemos tan clara la solución, que andamos como más perdidos, no más perdidos, sino ... yo creo que es que en un problema técnico todo el mundo tiene su idea de cuál es la mejor solución, pero no nos cuesta tanto entender la solución del otro e incluso admitir que aunque no sea la que nos guste más tiene sus méritos y es aceptable, o no nos es aceptable, pero nos cuesta menos bajar de lo nuestro, de donde estamos nosotros, acercarnos, echar un vistazo a lo del otro y forjarnos una opinión y decir, bueno, vale y tal ... En los problemas sociales yo creo que somos más intransigentes, nosotros tenemos nuestra idea de las cosas y todo lo demás está equivocado, yo creo que eso tiene algo que ver. Y sí, son los problemas que más tardamos en solucionar y que dejamos sin resolver porque no sabemos qué hacer con ellos, no nos ponemos de acuerdo y nadie sabe cómo salir. En ese sentido se ha hablado algunas veces de crear un [Comité Social], igual que tenemos un Comité Técnico, que resuelve las disputas técnicas entre desarrolladores cuando no se ponen de acuerdo en algo, como último recurso vamos al comité técnico, cada parte presenta sus argumentaciones y ellos toman una decisión y esa decisión es final. Y no tenemos lo mismo para las disputas sociales, o mejor dicho, para las disputas no técnicas. Y se ha dicho de crearlo pero no ha llegado a ninguna parte, porque su creación no es un problema técnico, es un problema social y no hemos llegado a un acuerdo y etcétera, etcétera.

Es decir, aquí se entiende lo técnico como aquello que se puede discutir fácilmente, y tiene, posiblemente por ello, una fácil solución. Bubulle añade a esta idea el hecho de que en Debian se piensa más como un ingeniero que como un especialista en relaciones humanas:

> **bubulle:** I think we are first a technical community, even if our cement are our values, we [...] these values with technical abilities and most of the people in the Project are technical people and this is sometimes one of our weaknesses, because our first reaction to problems is always an engineer reaction, is try to find a solution and for this, try to find the technical way to do things. We were talking about these levels of Debian Developer, Debian Maintainer, etc., and when this discussion happened, this was typical engineer discussion, how will this Debian Maintainer vote or whatever, and we bring a technical solution, which is one of our weaknesses, we are missing people more involved in human relationships. This is, probably I think in some way, this is one of the weaknesses of free software, because in the non-free, commercial environments, HP, Microsoft, etc., there are not only engineers, you have accountants, human resources people, all these things together to manage your company, and a free software project is made of engineers. Sometimes that is one of our weaknesses because we try to do this, but we are not specialists on human relationships.

Como indica dondelelcaro, lo técnico se entiende también como lo objetivo, lo que se plasma e implementa en los objetos que produce Debian:

> **dondelelcaro:** [...] hard enough time doing the technical things which are measurable relatively objectively [...] So at some layer yes, it's all what we could describe as a technical problem, but I think that, for me anyway, the main thing is, is it an issue that involves a change to things that Debian distributes?, or things that are on the machines that Debian runs? Those for me anyway are technical issues. Anything else is a social issue, generally, I mean. [...] If we hadn't the social aspect

we wouldn't do anything in Debian. They are sort of areas but they do overlap, and they certainly influence each other.

Agi introduce otra línea de definición. Lo social tiene que ver fundamentalmente con la comunidad con la que se comparte, con las relaciones basadas en el compartir y la reciprocidad:

agi: Yo creo que la parte social es fundamental. Es lo que hace que esto se mueva, porque por muy buenos técnicos que haya, si están en una empresa donde su desarrollo y su trabajo se queda dentro, pues no sirve de nada. Para mí la parte técnica es interesante porque es con la que me gano la vida, porque es la que me mola, me hace querer aprender, pero desde luego sin la social todo esto no sería posible, sin un grupo de personas que quieren compartir el conocimiento, que quieren ayudar al prójimo sin ánimo de lucro, sin eso no habría nada del resto. Entonces sin duda para mí la parte importante es la social, aunque a mí lo que me mole sea Linux, la red, el sistema operativo, el conocimiento, pero eso no existiría si yo no pusiera un poco de mi parte para que otra gente se beneficie, y si yo no me beneficiase de la generosidad de otros.

[...] Porque Debian parece que es sólo para gente rara y sólo para gente que sabe mucho. Es verdad que es una comunidad formada por técnicos muy técnicos, y a alguien que empieza pues le asusta, y es verdad que mucha de la gente que participa en Debian no tiene unas aptitudes sociales extremadamente desarrolladas, no son buenos tratando con personas, son buenos tratando con ordenadores.

Para gwolf, lo social se encuentra sobre todo en las relaciones interpersonales entre gente que se conoce:

gwolf: Pues mira, algo que yo siento que compartimos muchas personas es el que, y lo has visto a lo largo de estos días, aunque estamos

> haciendo trabajo técnico buena parte del día y el producto de Debian es un sistema operativo, pues parte del énfasis importante en una DebConf es la parte social. Hay relaciones interpersonales muy arraigadas ya, en ese sentido no veo nada que limite hablar de comunidad. Sí, es un grupo social antes que otra cosa.
>
> [...] el tipo de impacto que tendría. Algo técnico es algo que tiene una solución inequívoca, puede haber debate, claro, técnicamente; un proceso social es el que involucra el cómo vamos a trabajar, el quién va a trabajar, el cómo nos vamos a organizar, si vamos a cambiar alguno de los documentos fundacionales o cómo se ha de interpretar algo.

Luk indica que, en lo técnico, cualquiera llegaría a la misma solución. Es aquello en lo que todos son iguales, en tanto que tiene que ver con el conocimiento. Por el contrario, lo social tendría más que ver con la perspectiva particular de cada uno:

> **luk:** The big difference is that when you have a technical problem, actually everyone can solve it, because if you learn a bit about [...] the problem [...] you get the knowledge, and you can solve it, [...] the social problem means that some persons cannot get along with each other, something like that, and then even if you know the problem, even if you know the details of the problem, you cannot solve it, because it's still the problem between the persons, and the persons have to solve it. You can of course try to mediate, but that's not easy, not easy at all, because then you have to know the viewpoints of each of the persons, not only the viewpoints but also what binds them together and what are the actual problem points [...]. And that's a social problem, but if you don't meet the person, if you don't speak with the person, you can still work together, you can still make sure that your part of the job is done, without thinking about the other person, and the other person can use your work.

Por su parte zack, además de recoger algunas concepciones que han salido

antes, entiende que lo social se refiere a las formas de interacción humana, y los problemas sociales a las dificultades de esa interacción, poniendo por otra parte lo técnico en el centro del Proyecto:

> **zack:** For me the social problem in Debian is more related to how we interact with us and with others, so to me the word social problem evokes how we reply to each other on mailing lists, [...] I use the term non-welcoming community, that's the point, that is what I would call the centre of the social problem of Debian. The technical problem, [...], it's kind of weird because, we do have technical problems, I mean no software will be perfect, we maybe are lagging behind some other distribution, all of these are technical problems, sometimes we mix the two because we have, [...]. It is clear if you look at the founding documents of Debian, the Constitution, the Debian Free Software Guidelines, this kind of stuff, that the focus in the Constitution is purely technical, so the only arbitration device we have is the Technical Committee, that is the committee that can decide on any technical matter, on the assumption that all the possible matters of Debian are technical, [...], there is another [...] assumption that the only problem you can handle is a technical problem. So the arbitration device we have is the Technical Committee, which is responsible to decide only on technical matters, so as soon as a problem is not technical but it is social, or even borderline, we are not really well equipped in dealing with that.

Para mvz, lo técnico y lo social implican diferentes procesos de decisión. Pero lo más interesante es que muestra con claridad la dificultad de establecer y precisar esta distinción, mostrando cómo ambas dimensiones aparecen siempre mezcladas:

> **mvz:** Yes, but then, you know, [...] the former Project Secretary, he always says, in a few discussions I remember him saying, he is also member of the Technical Committee, once we start voting, once we start deciding technical questions by vote we are making a mistake,

> [...], you do not solve technical stuff by voting, so I would say, in that case we should [...] solve it in a different way than to vote. But if it does come to a vote, I would like everyone to be able to express their preference.
>
> [Ante una pregunta sobre la distinción entre los aspectos sociales y los técnicos del proyecto] I don't have an example ready so I have to construct one. A technical thing would be for example deciding, let's say for some reason one of our upstream projects turns into something really weird, they get involved with patenting stuff, they change the license, say, it's a ridiculous example but say the kernel Linux changes license, and many people in the project would not consider it free anymore, so it might come to making a decision about switching the default kernel to be kFreeBSD or something, that would ... Ah, actually it's a difficult question, it's both, it's both social and technical. Ok, let me try again, it's difficult, it really has both sides, it is not as easy to split of, ...
>
> [...] I'm right now, while I think about it, I find very difficult to make the distinction between the technical side and the social side, it's, I don't remember any vote which was clearly just social or just technical. Membership stuff is probably more on the social side of it, how do we handle members of the community which are causing, which are making [...] the work less fun for other people [...] sort of disrupting the project, that is pretty social, it's pretty clear a social thing. On the other hand, stuff like, no, I can't think of a genuinely technical example, it's not possible, I don't think there are any genuinely technical decisions. Even stuff like, how do we handle, for example, the fact that there are software patents in the US [...] we might be forced some day to sort of split the archive into a part of US and non-US, like it used to be for the crypto export stuff, so we will have to split it into software patenting countries and not patenting countries. That is sort of a technical problem in a way but it is also very strongly a social thing. How do we react to this, do we give in, and stop distributing the software in the US, or do we fight for it and wait to get into problems. Ah, interesting ...

Pero ésta no es sólo una distinción presente en el discurso privado, en el contexto de una entrevista, de los miembros de Debian. También aparece en documentos que reflejan discusiones en el seno del Proyecto. Como muestra, reproduzco aquí (véase cuadro 2.6, Discussions After Lenny; la página referida contiene enlaces a los documentos relevantes para la discusión) el texto de una página del *wiki* del Proyecto (Wiki.debian.org, s.f.[g]), que trata precisamente de mostrar los temas que se piensan merecedores de una discusión amplia dentro del Proyecto tras la salida de la versión *Lenny* de la distribución, que tuvo lugar el 14 de febrero de 2009. Lo importante aquí, más que los puntos concretos que aparecen, es el hecho de que estos aparecen categorizados en dos grandes apartados, *Organisational/Human* y *Technical*, que corresponderían a las dos categorías que estamos discutiendo. Esta clasificación se corresponde con el hecho de que, como indicaba antes, la categoría dominante de la oposición es la de «técnico», siendo la otra considerablemente más vaga, y pudiendo corresponderse con los términos de «social», «organizacional», o «humano». En cualquier caso, los temas clasificados como *Organisational/Human* aparecen en otros lugares como sociales, el término más utilizado.

En todos estos fragmentos citados observamos cómo, efectivamente, el polo dominante de la oposición es lo que se denomina como dimensión «técnica», y los problemas que se plantean en esta dimensión aparecen con perfiles más nítidos: «estamos haciendo trabajo técnico buena parte del día y el producto de Debian es un sistema operativo»; «a technical problem [...] is an issue that involves a change to things that Debian distributes, or things that are on the machines that Debian runs». E incluso en cierto sentido como *no-problemas*: «todos esos en los que somos muy competentes [...] y es muy fácil resolver»; «algo técnico es algo que tiene una solución inequívoca»; «actually everyone can solve it»; «the technical things which are measurable relatively objectively». En definitiva, la dimensión técnica se refiere a aquello que el Proyecto Debian produce, un sistema operativo construido como distribución de paquetes de *software*.

Por el contrario, el sentido de la dimensión «social» aparece más difuminado. De hecho, en el contexto de las entrevistas se usan diferentes términos como equivalentes a «social» cuando se plantea la oposición técnico/social: filosófico, político, interpersonal, valores, etc. El ámbito de lo social aparece como abierto y

This page tries to contain the list of things that should be discussed regarding the Debian's project, but should be delayed until after Lenny is out to avoid creating new problems for the release, and also to be able to discuss them in a calmer environment than that before a release.

Organisational / Human

- Membership classes and/or processes.
  - Discussion Starter: mjj29
- Having a Code of Conduct (CoC) in Debian. References for inspiration can be found in: http://www.ubuntu.com/community/conduct, http://www.kde.org/code-of-conduct/, http://live.gnome.org/CodeOfConduct, http://www.debian.org/MailingLists/, http://www.wesnoth.org/wiki/MP_CodeOfConduct, http://people.debian.org/~enrico/dcg/
- Restrict time people stay in positions of power, rotate people in these positions instead.
- Make the terms of DPL delegations written.

Technical

- Automated modification of debian/control files in cross-building and multiarch packages (gcc and glibc).
  - README.Debian
  - Describe or formalise some typical entries, like in manpages (Introduction, Quick-start, where-is-the-doc Howto, Enabling features (aka why install recommends/suggest).
- Use pseudo structured format, like asciidoc or rst. (An later, publish them on the web in p.d.o)
- Proposals/CopyrightFormat
- Proposals/DebianMenuUsingDesktopEntries (volunteer to drive the discussion and the transition: CharlesPlessy)
- switch away from defoma towards fontconfig (since it is cross-distro)

Cuadro 2.6: *Discussions After Lenny*

agonístico, en el que no existe una solución mejor por sí misma. Este ámbito se refiere al ámbito de las interacciones entre las personas y los problemas que implica, en los que los miembros de Debian se entienden poco hábiles: «mucha de la gente que participa en Debian no tiene unas aptitudes sociales extremadamente desarrolladas, no son buenos tratando con personas»; «we are not specialists on human relationships»; «the social problem means that some persons cannot get along with each other»; «for me the social problem in Debian is more related to how we interact with us and with others». Pero también se refiere a la filosofía del *software* libre y los principios de valoración que se asumen: «Yo creo que la parte social es fundamental [...] sin un grupo de personas que quieren compartir el conocimiento, que quieren ayudar al prójimo sin ánimo de lucro, sin eso no habría nada del resto»; «even if our cement are our values»; «it's social in that it affects our perceptions of things». O ambos aspectos a la vez: «un proceso social es el que involucra [...] el cómo nos vamos a organizar, si vamos a cambiar alguno de los documentos fundacionales o cómo se ha de interpretar algo». De hecho, la definición más clara quizás sea la negativa: lo «social» es aquello que no es «técnico»: «[...] las disputas sociales, o mejor dicho, para las disputas no técnicas»; «Anything else is a social issue»; y aquello que no es fácil de resolver: «Y luego están todos los demás, [...], que no tenemos tan clara la solución, que andamos como más perdidos». O como se expresa claramente en un mensaje a una lista de correo (Debian-project, 2009a), sobre la posibilidad de rechazar la integración en la distribución de paquetes cuando existe una relación problemática con el autor original del *software*:[13]

> Yes, this makes it difficult and lotsa work. But heck, we are going to reject packages based on social and **NOT** technical standards. If that gets to be an easy thing we are doing it wrong. Especially as the perception of social standard and behaviour tends to be very different From person to person, so having this be done by a large group hopefully makes it better, as more viewpoints are added.

En cualquier caso, la distinción ni es rígida, ni se plantea en términos exclu-

[13] El problema concreto al que se refiere se volverá a tratar en las secciones 4.6.1, *Debian como* Proxy. Upstream *y* downstream, y 5.3.2, *Organizar la colaboración ¿Existe la obligación de recibir en Debian?*

yentes. Antes al contrario, ambas dimensiones se piensan como necesariamente entrelazadas y en ocasiones como imposibles de separar: «it's difficult to reconcile them, so, actually it's difficult to tell whether it is social or technical, in a way, again, it's both [...] I'm right now, while I think about it, I find very difficult to make the distinction between the technical side and the social side, it's, I don't remember any vote which was clearly just social or just technical»; «They are sort of areas but they do overlap, and they certainly influence each other». Volveremos sobre esta cuestión más adelante.

Por otra parte, cuando en las entrevistas los miembros de Debian explican su proceso de ingreso en el Proyecto y sus motivaciones para participar en Debian, citan como razones de su atracción por Debian tanto la excelencia técnica de la distribución, como el compromiso con la comunidad y los principios del *software* libre, así como la importancia que tuvo para ellos en este proceso el establecimiento de relaciones sociales con otros miembros de la comunidad. Pero no los citan como motivaciones independientes. En gran medida, es ese aspecto «social» el que posibilita la excelencia técnica de la distribución ofrecida: «[...] Linux, la red, el sistema operativo, el conocimiento, pero eso no existiría si yo no pusiera un poco de mi parte para que otra gente se beneficie, y si yo no me beneficiase de la generosidad de otros».

Pero uno de los elementos más importantes de esta pareja de categorías *emic* es el hecho de que en su utilización, incluso si no siempre está bien definida y su uso no es consistente en diferentes contextos, se manifiesta un rasgo fundamental de la concepción que los miembros de Debian tienen de sí mismos y de su actividad: su autodefinición como «especialistas técnicos», como participantes en una forma de vida eminentemente tecnológica: «un problema técnico es [...] estamos en nuestra salsa»; «es una comunidad formada por técnicos muy técnicos»; «I think we are first a technical community [...] and most of the people in the Project are technical people». Esto es consistente con la concepción de la tecnología como un medio *cuasi*-transparente en el que los problemas no son nunca irresolubles, y sólo necesitan para su solución de mayor *expertise* técnica. Las dificultades reales provienen siempre de un ámbito más opaco (el de las relaciones humanas y sus motivaciones), más difícil de penetrar y regular por el tipo de saber por el que se autodefinen, incluso si es este ámbito el que posibilita e impulsa la creación de

artefactos y soluciones tecnológicas. De ahí el peligro percibido de tratar los problemas de este segundo ámbito mediante el tipo de racionalidad apropiada para solucionar los problemas técnicos: «our first reaction to problems is always an engineer reaction». Una tendencia que se encuentra incluso en los documentos fundacionales del Proyecto Debian: «it is clear if you look at the founding documents of Debian, the Constitution, the DFSG, this kind of stuff, that the focus in the constitution is purely technical, [...], on the assumption that all the possible matters of Debian are technical, [...], the only problem you can handle is a technical problem. So the arbitration device we have is the Technical Committee, which is responsible to decide only in technical matters, so as soon as a problem is not technical but it is social, or even borderline, we are not really well equipped in dealing with that».

Un Desarrollador centraba esta cuestión insistiendo en la idea de que lo técnico presupone un lenguaje compartido entre los miembros de Debian, poniendo las diferencias culturales y convencionales en el ámbito de lo social:

> **sto:** [A propósito de las discusiones sobre aspectos sociales] Son discusiones como las que puede haber en cualquier otro ámbito, lo que pasa es que aquí se convierten en mucho más peligrosas en el momento en que cuentas con que tienes gente de todo el mundo, con idiomas distintos, que hay problemas de comunicación a todos los niveles. Mientras estamos en lo técnico todos tenemos un lenguaje común, porque al final todos trabajamos sobre lo mismo y tenemos conocimientos parecidos en cualquier parte del mundo. Pero cuando nos salimos de ahí cada uno tiene su cultura y las diferencias a veces provocan unas discusiones interminables.

Sin embargo, y a pesar de esto, también es cierto que, aunque está muy presente la idea de que los problemas técnicos pueden resolverse de manera objetiva y única, en numerosas ocasiones una solución técnica ha de ser elegida entre varias posibles, con consecuencias diferentes para el trabajo de otros miembros de Debian. En estos casos, se elegirá normalmente ejerciendo la meritocracia o la doocracia (véase la sección 3.2, *Toma de decisiones: de la meritocracia a la doo-*

cracia). En cualquier caso, un elemento que relativiza esta dicotomía es la alta valoración en ambos ámbitos, el técnico y el social, de la diversidad y la heterogeneidad por parte de los miembros de Debian. En cuanto a la diversidad en el ámbito social, véase la sección 6.3.3, *Diversidad en Debian: otros procesos de reforma*. En Debian cada Desarrollador puede elegir las herramientas técnicas que considere más adecuadas, y se da de hecho una gran diversidad al respecto. A propósito precisamente de lo ocurrido con `python` decía un Desarrollador:

> **sto:** Se implementaron dos mecanismos para soportar el empaquetado de programas en `python`. Aquí en Debian hay una cosa que es muy buena. Se plantean políticas de funcionamiento de distintas cosas, por ejemplo cómo se gestionan los programas en `python`. Se define una política de cómo se tienen que gestionar. Cuando se ha definido la política se buscan herramientas técnicas que garanticen, o sea que simplifiquen seguir la política, y que garanticen que se detectan las cosas que no siguen la política. [...]
>
> En `python` se plantearon dos maneras de resolver un problema, que era las transiciones entre una versión del intérprete y otra. Para evitar como se hacía en `perl`, recompilar todos los paquetes para que usaran la nueva versión del intérprete y subirlos al archivo. Era farragoso. [...]
>
> Hoy en día [2009] siguen en uso dos herramientas, hacen cosas muy parecidas para hacer lo mismo. ¿Quién gana? Igual se mantienen para siempre las dos. No hay nada de malo, si las dos tienen un caso de uso y tiene sentido mantenerlas. Las discusiones técnicas a veces se solucionan dejando la puerta abierta a más opciones.

Por último, señalaremos que esta distinción es también contextual en el sentido de que definirse como técnico supone establecer implícitamente un campo específico de especialización técnica. Aunque su campo de especialización suele de hecho ser identificado como lo tecnológico sin más o al menos como lo tecnológico por excelencia, los Desarrolladores de Debian son conscientes de esta cuestión:

> **marga:** Ejemplos de cuestiones sociales son digamos, que sé yo, todo

> lo que tiene que ver con la filosofía del *software* libre. Por ejemplo si la GPL v. 3 cumple con las DFSG o no. Eso es más bien una decisión social y no técnica. [...] Es una cuestión legal básicamente, o sea es técnica de los abogados [risas]. No es técnica de lo nuestro, es técnica de los abogados.

2.2.1.1 La posibilidad de establecer un Comité Social y el Código de Conducta.

Una situación en la que se observa muy claramente cómo se establece la distinción entre estos dos ámbitos de lo técnico y lo social es en el diferente tratamiento institucional que reciben los conflictos que se plantean en ambos ámbitos. En parte como intento de solucionar esos problemas de interacción personal, en ocasiones se ha propuesto en el seno del Proyecto Debian la necesidad de establecer un Código de Conducta,[14] o de llegar a un acuerdo en torno a las formas adecuadas de comunicación e interacción.[15] Así, en enero de 2007 encontramos una discusión en la lista `debian-project` (Debian-project, 2007b) sobre la creación de un «Comité Social», que finalmente no prosperó. No se trata aquí de analizar la discusión en profundidad,[16] pero sí que me interesa destacar que el papel y el funcionamiento de este Comité se piensan según el modelo del Comité Técnico. En la propuesta encontramos que ésta se modela según lo que la Constitución regula sobre el Comité Técnico:

> Comparing with section 6., Technical committee, let's see:
>
> > Decide on any matter of technical policy. This includes the contents of the technical policy manuals, developers' reference materials, example packages and the behaviour of non-experimental package building tools.
>
> This one could be tricky to phrase. Maybe - «Decide on any social matter, including social norms and customs, non-technical communica-

[14] A semejanza del «Código de Conducta de Ubuntu», una distribución derivada de Debian (Ubuntu.com, s.f.[b]).

[15] Como las *Debian Community Guidelines* desarrolladas por el Desarrollador de Debian Enrico Zini (Zini, 2006).

[16] Se puede encontrar un buen resumen en Schulze (2007).

> tion among developers, and day-to-day organization matters within the Project.»
>
> The points 6.1.2. through 6.1.7. should all be included, with s/technical/social/g or so.
>
> > The Chairman can stand in for the Leader, together with the Secretary As detailed in §7.1(2), the Chairman of the Technical Committee and the Project Secretary may together stand in for the Leader if there is no Leader.

La expresión `s/technical/social/g` indica que se debe sustituir el término «technical» por «social» cada vez que aparezca. En uno de los mensajes en el hilo se puede leer:

> When the subject was first bruited in in the shadowy secrecy of -private;[17] it was associated with the technical committee. it was said that the social committee will be like the technical committee, except for social and cultural issues. Like the tech ctte, it would make policy, `_social policy_`, it would define the norms, and so on. The constitutional bits related to the DPL and tech ctte were quoted as a model for the social committee.
>
> And the technical committee is the highest authority on technical issues in the various institutions the constitution delineates.

Es decir, de nuevo observamos que el polo dominante en la autoconcepción de los miembros de Debian respecto a su actividad es el técnico. Pero si la resolución de los problemas llamados «sociales» se piensa según el modelo de los problemas «técnicos», encontramos también una desconfianza ante la posibilidad de aplicar este tipo de resolución de problemas a un ámbito que se entiende en cierto modo externo a la actividad técnica que constituye la razón de ser del Proyecto Debian:

[17] Se refiere a la lista de correo `debian-private`, una lista a la que sólo tienen acceso los Desarrolladores de Debian.

> **dondelelcaro:** If it could work, I wouldn't have a problem with it, my concern is that it won't, the reason why is, [...] people in Debian have a [...] distrust of rules and regulations, especially when it comes to social conduct, which is such an imprecise [...], which is sort of hard because, one of the thing that happens in Debian is we have policy, we have a lot of things that have very special rules, but other things can't be understood technically. [...] but for social interactions, there are always gray areas, and it is very difficult for people to identify that [...] We want people to identify the problems they have in communicating, and have tools to fix the communication, have a group of people who are able to approach and help out when the communication goes wrong. [...] We don't need a Code of Conduct, we need like guidelines of conduct, [...] a cookbook for when your interactions go wrong.

Finalmente, en abril de 2014 se aprobó mediante una Resolución General (puede verse en Debian.org/vote (2014)), una votación de todos los Desarrolladores Debian, un Código de Conducta (Debian.org, s.f.[k]). Pero efectivamente son más unas guías generales que un código estricto, y su ámbito de aplicación se refiere a los medios de comunicación del Proyecto, como listas de correo y canales de IRC. Se establece la posibilidad de expulsar a quien no lo cumpla de los canales de comunicación señalados, pero la decisión corresponde a los responsables de esos canales.[18]

2.2.2 La distinción entre lo técnico y lo social en el *bug* sobre el mantenimiento de Python.

Volvamos por un momento a la historia sobre el *bug* relativo al mantenimiento de `python`. Como hemos visto, los problemas «técnicos» de los que se ocupa el comité están generalmente mezclados con problemas de comunicación entre las diferentes partes implicadas, lo que tiene como consecuencia que la distinción entre ambos ámbitos no sea clara, y esté abierta a la interpretación y la negociación. Hemos visto también en las citas cómo los miembros de Debian dudan y vacilan en el momento de ofrecer una definición de la distinción, y sobre todo al

[18] Para una controversia sobre la posibilidad y dificultad de su aplicación, véase Corbet (2014).

asignar casos y problemas concretos a uno de los dos ámbitos. Esto incluye también a los implicados en la discusión sobre el mantenimiento de `python`. En una entrevista me decía uno de los miembros del Comité Técnico a propósito de este problema:

> **dondelelcaro:** That one is complicated, so underlying this problem is actually a technical issue, and so, the technical issue is precisely how to handle python modules installation [...] that literally is the core of the technical problem. Now, layered on top of that, there is a communication issue where one party has decided that the communication is no longer worth to bother, in the sense that there have been attacks going, and it is just non productive to communicate with certain people.

Y el Debian Project Leader añadía:

> **zack:** That is a good example because there are both issues, in fact there are two problems. One problem is, a basic technical choice of which tool to use to package python applications, so there is a technical problem on that matter which has not been resolved. But this is not the problem which has been brought to the attention of the Technical Committee. The problem which has been brought to the attention of the Technical Committee is, who is the maintainer of the python package. Well, there is one maintainer, but people were asking the Technical Committee to overrule who is the maintainer of the package, and assign the package to someone else. That is what has been asked to the Technical Committee, which has not been solved yet, the bug is still outstanding. So one can ask, so there is a social problem in there because there is a communication problem between the maintainer and all the rest of the python community. So this is something not even the maintainer would deny. That is something that you can say is a social problem, because it is a problem in the interaction among people. But it's been kind of reinstated into a technical problem, and asked to the

> Technical Committee, ok, we ask you to decide who is responsible for the package. So, yes, it can be seen as a …
>
> **Fernando:** The Technical Committee can do that? Decide who is …
>
> **zack:** That's a good question, and I was asking myself the very same thing while I was saying that, [risas] I don't have in mind exactly the Constitution right now, I think one can argue that they can do that because the area of responsibilities in Debian is something that the Constitution is aware of, so on the assumption we were making before that the Constitution is on […] technical matter, well, then yes. One might even say that the technical decision is what to write in a specific file which is «debian/control»,[19] so yes you can say that is a technical problem.

Curiosamente, unos meses después de la entrevista el mismo DPL hacía una propuesta (que no prosperó) en el BTS que presuponía que la decisión sobe el mantenedor es social y no técnica (Bugs.debian.org, s.f.[e]):

> As I feel bad about not having anything better to do than ping you, here comes my last attempt to push this topic forward and, hopefully, get it fixed within the current DPL term (of course, such a political concern is not a CTTE concern, so you are more than free to ignore it).
>
> The proposal is to delegate the decision on Python interpreter maintenance to the DPL. No technical decision would be part of the delegation, only the «social» problem of who are the maintainers will be. There seem to be no specific provision in the Constitution about the CTTE delegating decisions to others, but if the CTTE agrees on that (ideally, voting on it), I think we'll be formally fine.

Aquí vemos que el Comité Técnico no debe ocuparse en principio de cuestiones sociales o políticas, ni el DPL de cuestiones técnicas. Como se ve en el cuadro

[19] Es decir, escribir en un archivo, llamado `debian/control`, del paquete fuente el nombre del mantenedor del paquete. Véase la sección 4.5, *El empaquetado como práctica sociotécnica*.

2.2, *Algunos poderes del Comité Técnico*, el punto 6.1.2 de la Constitución establece específicamente que sí entra dentro de los poderes del Comité Técnico decidir quién es el mantenedor de un paquete. Pero incluso aunque no fuese así, vemos cómo sería posible que pudiese decidir quién es efectivamente el mantenedor, si el mismo comité *decide* que es un problema técnico. Lo destacable es que es algo que se debe decidir.

Y es que, como ya hemos afirmado anteriormente, la asignación de un problema al campo de lo social o de lo técnico ha de ser decidida y construida en cada ocasión. Esto es manifiesto en la discusión tratada. En realidad, esta controversia muestra con claridad que lo social es el residuo que queda, de ahí que se intente desde el principio delimitar qué es lo técnico del asunto. En un mensaje del 22 de marzo de 2010 (Debian-ctte, 2010a), uno de los firmantes del informe de fallo contesta a un mensaje anterior de un miembro del Comité Técnico, citando algunos fragmentos:

> It bothers me that what you've brought to the TC is a rant about your
> frustrations culminating in a request to remove someone else from
> their role, rather than a crisper articulation of what's wrong and a
> plan that explains how we should move forward.

We are sorry if it sounded that way, because this is not the message we wanted to convey. We tried to list the current issues from a technical point of view, and if you want us to elaborate on a specific point, we'll be happy to provide more input. Overall, our request indeed lacks a complete plan of what we want to implement; because it is not about what should be implemented, but about how it should be discussed, elaborated and coordinated. That said, we understand that you want us to explain what we would propose and implement if we were maintainers for Python, so we will try to elaborate on that. It is not that easy because we have no clear idea of what the current plans of the Python maintainer are.

[...]

> I realize this may not be what you had in mind when you asked the TC

> to «decide about how the Python interpreter packages should be

> maintained in Debian«, but now that you've opened Pandora's box, I

> believe we have a responsibility to understand the underlying

> problems that apparently have plagued Python policy for years, so

> that whatever decision we take will ensure the most positive outcome

> for Python handling in Debian in the future.

This is somehow reassuring that the technical committee tries to understand underlying technical issues. Thank you for making this technical and trying to resolve this situation, that's really appreciated.

Lo que en el informe de fallo inicial no aparecía con claridad, la cuestión de si estamos ante un problema social o un problema técnico, se va perfilando poco a poco en los hilos en los que se produce el debate. Ahí, cada una de las partes intenta mostrar cuál es según su interpretación el elemento técnico a reconfigurar para conseguir la resolución del problema. En lo que todos están de acuerdo es en que aislar ese elemento técnico es lo que permitiría continuar la discusión y que el Comité Técnico encontrara una solución. En un mensaje de mayo de 2010 (Debian-ctte, 2010b) vemos cómo uno de los proponentes reafirma que, más allá de otros problemas que pueda haber, sus quejas se refieren a cuestiones técnicas:

> However, there have been some talks within the tech ctte and with different

> people inside the Debian python community. That needed time, and we prefer

> to get to a resolution a bit later than to one that doesn't work. I doubt

> we can get to a final decision as of now.

> The complaints are mostly non-technical. Re the technical parts, e.g. the
>
> discussion within Debian about «where to store files» brought two upstream
>
> proposals (PEPs) which would fix some disagreements in an good and
>
> forward way. I don't want to loose Matthias contributions to python
>
> within Debian and the python community.

Please let us reaffirm our «complains» are also technical: try not put them aside just dealing with the personal situations around Python. In several occasions (even recently) there were situations we can mention to challenge the «technique» (either being the pure packaging technicalities, or the attention/interest dedicated, or more); we don't want to point fingers directly to those mistakes, but let's not reduce this techctte call to only feelings.

Casi un año después, el mensaje de uno de los miembros del Comité Técnico muestra claramente que esa distinción no acaba de estar claramente delimitada, o no del mismo modo para todos los implicados (Debian-ctte, 2011b):

But you are again conflating the social and the technical, where I'm trying to draw a distinction. The **technical** problem that needs to be solved is that for over half a decade we've been without a consistent, agreed-upon description of how python packages are supposed to interact in Debian, and as a result there are packages working at cross-purposes. That Matthias is not the one doing the work to fix this is only relevant to the **social** problem, the lack of trust and collaboration between the python maintainer and the python modules maintainers which is largely secondary to the technical one (although it certainly has legs of its own by now). Solving the technical problem would open the door to improved collaboration (and also clear the air so that the TC could rule on the maintainership question **per se** if there were still

> problems with the interactions between python and modules maintainers). In contrast, addressing the social problem by kicking Matthias out leaves the technical issues unanswered.

Según lo visto antes sobre las interacciones sociales en Debian, no es extraño que muchas veces resulte difícil distinguir entre ambos aspectos para los participantes. Un desarrollador de Debian definía así los problemas sociales que se dan en Debian:

> **gregoa:** Well, social has to do with people, obviously, so social ... problems, conflicts, disputes, fights, between people ..., no, more generally, problems that rise from a social interaction between people.

Problemas que surgen de una interacción social entre las personas. Parece claro, pero resulta que en el Proyecto Debian la mayoría de las interacciones sociales están mediadas por dispositivos tecnológicos. Y lo que es más importante, suelen estar relacionadas con la interacción de esas personas con esos u otros dispositivos tecnológicos. Otro desarrollador lo expresaba con mucha claridad:

> **clint:** Ok, well, a problem is social if it is about how people interact, and technical if it is about how software interacts. And then, if you have people interacting with software then maybe it gets confused.

2.2.3 La producción práctica de la distinción.

Así pues, ¿qué es lo que están haciendo los miembros de Debian al distinguir entre los aspectos técnicos y los sociales, cuando *producen* esta clasificación en relación a una determinada controversia, como por ejemplo la que vimos en la sección 2.1, *Bug #573745*? No hay que dar por supuesta la validez de este esquema para interpretar y clasificar las prácticas y procesos que configuran el campo social de la investigación. Quizás el peligro más inmediato sea el de ceder a un dualismo rígido que tampoco se encuentra, como espero haber mostrado, en la utilización de la oposición *emic* analizada entre los miembros del Proyecto Debian. Es común en

los estudios sobre ciencia, tecnología y sociedad poner en cuestión la naturalidad de la distinción entre lo técnico y lo social, pero la cuestión es que los miembros de Debian la tematizan explícitamente, reintroduciendo esa distinción. Se trata entonces de considerar etnográficamente esos elementos técnicos, e indagar en la construcción de esa misma distinción. El cómo y el por qué lo hacen se convierte en problema de investigación, más que en marco donde situar los procesos.

Que la distinción entre lo técnico y lo social se va construyendo permanentemente se ve con claridad en el hecho de que los ejemplos no están casi nunca claramente delimitados en las entrevistas, donde no aparecen prácticamente respuestas ya preparadas, sino que se manifiesta el proceso de pensamiento. Pero no se construyen como representaciones o contenidos de conciencia *a posteriori*, sino como parte efectiva del proceso de resolución de problemas, como forma de delegación de acciones y agencia. Cuando un conjunto de prácticas se asigna a una dimensión técnica o a una dimensión social, también se está al mismo tiempo creando esas dimensiones a partir de ellas, articulándolas y dándoles un orden. Se les asigna una manera de resolverse y una forma de acción social adecuada. No es por lo tanto que se construya una representación, cuanto una forma de actuar y de organizar. Por otra parte, el hecho de que lo social es lo queda tras un proceso de elaboración, el resto, muestra claramente que esa distinción es producto de un proceso de separación, de purificación, de algo que en su origen está mezclado, es un híbrido (véase la sección 4.4, *El código como objeto híbrido*).

Entonces, cuando en un proceso de discusión se negocia si los puntos implicados se han de asignar a un campo técnico o a un campo social, lo que se está decidiendo es qué tipo de problema tienes enfrente: un objeto técnico que puedes dominar y poner a tu servicio, o personas que se te enfrentan y tienen diferentes concepciones. Y es una manera de efectuar el reparto de la agencia porque se está decidiendo quiénes son los actores relevantes, quiénes van a tener capacidad de decisión y de hacer prevalecer sus fines, y cuáles son las formas de resolución y de mediación adecuadas para el caso. Convencer a los demás de que lo técnico de una determinada cuestión está donde tú afirmas, supone que cualquiera solucionaría, al menos en principio, esa cuestión del mismo modo que tú.

Los mecanismos de coordinación y los procesos de toma de decisiones y re-

solución de conflictos son muy variados en Debian, como veremos en el capítulo siguiente. Las dos formas de toma de decisiones cuando existe algún conflicto que no puede ser superado de otra manera, o cuando afecta al Proyecto en su conjunto (y de hecho relativamente poco utilizados) son el Comité Técnico para la disputas técnicas y las *General Resolutions*, Resoluciones Generales, un proceso de votación en el que participan todos los Desarrolladores de Debian. Y si, como hemos visto, el Comité Técnico debe limitar su poder de decisión a los elementos técnicos y es considerada generalmente como una mala idea la de instaurar algo semejante a un Comité Social, algo parecido ocurre con las Resoluciones Generales: resolver una disputa técnica mediante una votación es usualmente considerado como una de las peores resoluciones posibles. En relación precisamente al problema que veremos en la sección siguiente sobre el *firmware*, decía un Desarrollador:

> **karora:** The thing that it has to do with, on the one hand, is very technical, because is talking about firmware. But the real question is a social one. It is about the freeness of the technical component. So it's a deeper social question, it's definitely a social question. And there is involved a General Resolution, and if you look through the General Resolutions, they are all social questions. The process of having a General Resolution is kind of the last resort for social questions. You wouldn't put a technical question at a GR, that cannot work.

La otra cara es la advertencia tan frecuentemente oída y con la que empezábamos esta sección: «no debemos resolver problemas sociales con soluciones técnicas». Esta máxima de no intentar resolver problemas sociales con medidas técnicas expresa el intento de separar esos dos aspectos. Lo que se pretende evitar es un cierre prematuro, en falso, del problema. Porque dar una solución técnica quiere decir inscribirla en un dispositivo tecnológico en lugar de dejarla abierta. Lo social sería aquello que está en proceso de definición, de articulación y rearticulación. Lo técnico haría referencia, por el contrario, a las formas de mediación y traducción que están estabilizadas, que no se problematizan en ese momento. Si lo social es lo que todavía no se puede implementar técnicamente, se entienden entonces las dificultades para hacer cumplir efectivamente el Código de Conducta.

El *firmware* es «la combinación de un dispositivo *hardware* y los datos e instrucciones de computadora que residen en ese dispositivo como *software* de sólo lectura», según la definición del *Institute of Electrical and Electronics Engineers* (IEEE) (IEEE Standards Information Network/IEEE Press, 2000, pág. 438). Es decir, los datos y/o pequeños programas que determinan el comportamiento de un dispositivo desde el interior del mismo, controlando sus operaciones de bajo nivel. Muchos dispositivos y periféricos deben cargar partes del *firmware* desde el ordenador, normalmente desde el *driver* del dispositivo. Y entonces, este *firmware* puede ser «libre» o «propietario», según la licencia que tenga (véase la sección 5.2.1, *Las licencias libres*). En el mundo del *software* libre, se denomina *binary blob* a tales partes cuando sólo se distribuyen en forma binaria, sin código fuente. Sobre esta cuestión y sus implicaciones para el *software* libre, véase Corbet (2005).

Cuadro 2.7: *Firmware*

2.2.4 Un ejemplo: el *Firmware* y la inscripción de programas de acción.

Veamos un caso en el que se ve claramente que la definición de un problema como social, y su tratamiento como tal, depende efectivamente de que su solución *todavía* no admite una mediación técnica. Son recurrentes las discusiones en el interior del Proyecto sobre la inclusión de *firmware* (ver Cuadro 2.7, *Firmware*) no libre en la distribución. Detengámonos un momento en esta cuestión.

Cada cierto tiempo, se produce una discusión en el seno del Proyecto Debian sobre la conveniencia o no de eliminar de la distribución las partes del *firmware* que no son libres y no van acompañadas del código fuente, lo que en principio violaría las *Debian Free Software Guidelines* (véase el apéndice B, *DFSG*). Como dice gráficamente un Desarrollador entrevistado: «Parece impepinable que antes de cada release haya una discusión sobre el *firmware* del *kernel*.» Efectivamente, antes de que las versiones 4.0 (*Etch*) y 5.0 (*Lenny*) del sistema operativo fueran publicadas, hubo sendas Resoluciones Generales[20] para determinar si se publicaban con *firmware* no libre, o si éste se excluía. En ambas ocasiones la opción

[20] Sobre las *General Resolutions* como mecanismo de decisión, véase la sección 3.2, *Toma de decisiones: de la meritocracia a la doocracia*.

vencedora fue la de permitir la publicación con *firmware* no libre. En la votación del 2006 la opción vencedora fue «Release Etch even with kernel firmware issues» (Debian.org/vote, 2006), y en la de diciembre de 2008 la opción ganadora fue «Assume blobs comply with GPL unless proven otherwise» (Debian.org/vote, 2008a). Ambos resultados suponen de hecho la inclusión de *firmware* no libre en la distribución.

Como se puede observar en un *blog post* de un Desarrollador de Debian (Williams, 2009), una de las claves de la polémica estuvo en la cuestión de si el *firmware* debe ser producido a partir de código fuente (en definitiva, si el *firmware* es *software*, y por lo tanto tiene sentido exigir que su código fuente esté disponible y sea libre), o son simplemente datos a los que no tiene sentido aplicar ese requerimiento: «Everyone seems hung up on firmware as something that requires source code, when the real point is that firmware isn't necessarily like other content in Debian and in some (many?) situations, there is no, nor can be any, human-readable form» (Williams, 2009). Además, casi todas las propuestas de resolución que aparecen en la votación argumentan su posición en base a los mismos documentos, fundamentalmente el Contrato Social (apéndice A, *Contrato social*). En lo que difieren, más que en la adhesión a diferentes principios de valoración (fundamentalmente, el compromiso con los usuarios y sus necesidades, lo que llevaría a incluir el *firmware*, y el compromiso con la libertad del *software*, distribuido, lo que llevaría a excluirlo), es en la interpretación de esos principios y en su importancia relativa en ese determinado contexto.

La cuestión es que ambos principios eran incompatibles en esa situación en concreto. Cumplir estrictamente con las DFSG impediría incluir en la distribución determinado *firmware* necesario para hacer funcionar dispositivos de numerosos usuarios. Por lo tanto, el Proyecto se encontraba ante un dilema. Pero, ¿estamos ante un problema técnico o un problema social? Pues no está claro. En una entrevista, cuando le pedía a un Desarrollador de Debian que me pusiese ejemplos de discusiones claramente técnicas o sociales, me respondió aludiendo precisamente a esta discusión:

luciano: Es que el problema no es lo que claramente es, el problema

> es lo que está en el medio. Por ejemplo, *blobs* binarios en el *kernel*. ¿Es técnico o social? Sólo tratar de distinguirlo ya es todo un problema. Uno diría, es técnico, vamos, estamos hablando de *blobs*, de binarios, de *kernel*, ... Sin embargo, si hace que una instalación sea menos *friendly* para un usuario final, se transforma en un problema social. Sólo clasificarlo ya es todo un problema. Si lo lograses clasificar, ya tendrías parte del problema solucionado. Uno diría, bueno, ok, es un problema técnico. Bueno, en ese caso, analicemos cuáles son los pros y los contras de incluir *blobs*.

Otro Desarrollador insistía en la idea de que es difícil establecer si es una cuestión técnica, social, o ambas:

> **mvz:** It was social, actually it was neither. It was more of a matter of, mmm, or maybe it was both. It's about how do we see the world. We decide to come to a common, to vote to get to a common view of how things are, how we perceive things. There are people who say, firmware is like an extension of hardware, we shouldn't care, it could be on the ROM, it doesn't matter if it is a file, we shouldn't care, it is not relevant for Debian, we should just ignore it. Then there are people who say it's software so we should care ... I think it was more of a, of, people still have different opinions about that to this date, ..., and you could [...] really heat the discussion if you get the right people together, so, it's difficult to... There're some good arguments to both sides, it's difficult to reconcile them, so, actually it's difficult to tell whether it is social or technical. In a way, again, it's both, it's technical in that we, mmm, the implications are technical, and it's social in that it affects our perception of things. So, if I vote and I'm on the losing side so I'm on, for example the opposite opinion is stronger, more people vote for it, it's a social thing because the project by majority, decides to take a point of view which is different than my personal point of view. I think that's social, it's a social thing, I think.

Sin embargo, uno de los miembros del Comité Técnico sí afirmaba con cla-

ridad que nos encontramos ante una discusión más social que técnica:

> **vorlon:** It is not a technical question, it is a political question, because if it were a technical question, there would be no reason not to let the maintainer make the decision. Because we expect maintainers to be able to make correct decisions [...] the software does what it is supposed to do, so it is a political issue. [...] The arguments are entirely political, they are about where we draw the line of what we consider free software.

Lo más interesante de esta última cita no es tanto que establezca claramente que es una cuestión social, sino que muestra claramente que establecer esa diferencia determina quién ha de tomar la decisión. Si fuese técnico, puede resolverlo el mantenedor en su papel de experto (o, en su caso, el Comité Técnico). Si no lo es, su resolución afecta al Proyecto en su conjunto:

> **karora:** You can find an expert when you have a technical problem. You can pick somebody and say, well that person knows what they are doing. Let them answer, let them make the decision. Very straightforward to delegate the resolution of such problems. It is much easier to argue them. If you have 2 or 3 people who are all experts, then there is a small encapsulated argument. If they respect each other's technical abilities, then they will just discuss and came up with a solution. And it may well be a solution better than any individual could have come up with. Really there is not any point in the wider community getting involved. Social problems are much harder because everybody is an expert in social problems.

Y la manera más clara de resolver un problema por parte del Proyecto en su conjunto, cuando no se puede recurrir al consenso informal, es mediante una Resolución General:

> **vorlon:** The General Resolution is a political decision-making process, as is the DPL, a political figure, and they are not supposed to make technical decisions. You can have a General Resolution to override a technical decision, but that would not be the preference, generally.

En cualquier caso, sólo decidiendo si estamos ante una cuestión técnica o social se puede determinar quién tiene el poder para resolverla, lo que por supuesto nunca es fácil:

> **vorlon:** Sometimes we have discussions about whether one is one or the other, which can be a challenge as well, and try to decide who has the decision making power. This is actually one of the gray areas within the Project.

En los momentos de la publicación de *Etch* y *Lenny* el dilema era irresoluble, manifestándose como un problema social en torno a los principios que debían prevalecer, y por lo tanto la manera de poner una solución provisional era mediante una Resolución General. Pero cuando llegó el momento de publicar la siguiente versión del sistema operativo, Debian 6.0 *Squeeze*, la situación había cambiado. De nuevo, no porque se hubiera discutido más extensamente sobre la conveniencia o no de incluir *blobs* binarios, o se hubiera llegado a un acuerdo sobre si el *firmware* es *software* y como tal se ve afectado por las *Debian Free Software Guidelines*. Sino porque por fin era posible ofrecer una mediación técnica para el dilema, de tal modo que se disolvía la tensión entre los dos principios anteriormente enfrentados. Ambos podían incorporarse, ser inscritos, al mismo tiempo en el dispositivo tecnológico que permite la instalación del sistema operativo en un ordenador. Básicamente, lo que se había conseguido era la separación de las partes libres y las no libres del *kernel* de Linux, por parte del *Debian kernel team* con la ayuda de los desarrolladores *upstream* del *kernel*, y la posibilidad de que el usuario cargase el *firmware* no libre en el momento de la instalación, gracias a las innovaciones introducidas por el *Debian-installer team* (Debian.org/News, 2010; Zacchiroli, 2010c). El *firmware* no libre se incluye en la sección *non-free* del archivo (véase la sección 5.2.3, *Las licencias en Debian*), lo que supone que no

es realmente parte de la distribución Debian. Si la decisión de eliminar el *firmware* no libre se hubiese tomado antes de que estos cambios en la infraestructura hubiesen estado listos, el usuario habría que averiguar qué *firmware* necesitaba, buscarlo por su cuenta e instalarlo de tal modo que quedase bien integrado en el sistema, sin la ayuda de las herramientas de manejo de paquetes de Debian y posiblemente con un ordenador no completamente funcional, si lo que faltaba era el *firmware* de la tarjeta de red o de la tarjeta gráfica. Estas tareas están más allá del alcance de la gran mayoría de usuarios.

De tal modo que *Squeeze* se publicó sin *firmware* no libre, sin que hubiera necesidad para ello de una nueva Resolución General. Sin mucho ruido, un problema social persistente había desaparecido gracias a la inscripción simultánea de dos programas de acción que antes eran incompatibles en el mismo dispositivo tecnológico. Según Latour:[21]

> The program of action is the set of written instructions that can be substituted by the analyst to any artifact. Now that computers exist, we are able to conceive of a text (a programming language) that is at once words and actions. How to do things with words and then turn words into things is now clear to any programmer (Latour, 1992, pág. 255).

Pero los programas de acción no se encuentran sólo en los dispositivos tecnológicos, sino que también son asumidos por los agentes humanos:

> Students of technology are never faced with people on the one hand and things on the other, they are faced with programs of action, sections of which are endowed to parts of humans, while other sections are entrusted to parts of nonhumans (Latour, 1992, pág. 254).

Efectivamente, en un determinado programa de acción se distribuyen competencias entre humanos y no humanos, artefactos o dispositivos tecnológicos.

[21] Véase también Akrich y Latour (1992, pág. 260).

Eso encontramos aquí, una nueva forma de compatibilizar los programas de acción centrados en facilitar la vida del usuario y los centrados en promover la libertad del *software*. Antes de *Squeeze* no distribuir *software* no libre hubiese requerido delegar en los usuarios una serie de acciones complejas para que pudiesen usar su *hardware*, cuya necesidad de usar sus dispositivos se habría convertido entonces en un antiprograma de acción (Akrich y Latour, 1992, pág. 261), en tanto que esas acciones irían en contra de la eliminación e intentarían contrarrestarla. A partir de *Squeeze*, las innovaciones tecnológicas introducidas pueden encarnar al mismo tiempo esos dos programas de acción.

Más que un proceso social en el que lo técnico y lo social se mezclan, lo que encontramos en este caso es un proceso de *separación* y *construcción* de estos dos ámbitos. En un primer momento, antes de la publicación de *Squeeze*, la imposibilidad de resolver tecnológicamente esta cuestión hizo que se desplazara a un ámbito que los miembros de Debian llaman «social». Un ámbito que es creado precisamente por ése y otros desplazamientos similares, dejando en el centro un ámbito de cuestiones en las que es posible seguir actuando a través de la mediación de dispositivos tecnológicos, y que los miembros de Debian llaman «técnico». En el proceso de desarrollo de *Squeeze* dejó de ser necesario ese momento de desplazamiento. Ambos ámbitos implican formas de acción social y prácticas diferentes. Esto no quiere decir en absoluto que todo lo que los miembros de Debian consideran un asunto «social» vaya a desaparecer en algún momento a causa de la creación de algún nuevo *software*, ni desde luego que ellos lo entiendan así.

Así pues estamos describiendo una separación de ámbitos que no sólo es conceptualizada, sino producida efectivamente por la acción social de los miembros de Debian, y producida a lo largo de toda una serie de controversias y disputas. La distinción tiene entonces como presupuesto una serie de *translations* (como vimos en la sección 2.1.7, *Procesos de traducción*), en su doble sentido de desplazamientos y traducciones. Se traduce el significado de las disputas a las formas de acción social de cada ámbito, se movilizan una serie de dispositivos (en el caso del *firmware* de manera bastante literal: la resolución de la controversia pasó por mover la localización de las partes no libres del *kernel*) y se constituyen nuevos agentes. Cobra ahora todo su sentido la afirmación de Latour que apareció anteriormente: «las operaciones de traducción transforman las cuestiones políticas en

cuestiones técnicas y viceversa». Esto es lo que hemos visto en las controversias sobre el mantenimiento de `python` y sobre el *firmware* no libre.

A primera vista, esto podría suponer una tendencia a intentar clasificar problemas como técnicos antes que sociales, porque serían más fáciles de solucionar. Un miembro del Comité Técnico lo expresa así:

> **vorlon:** We try to avoid having too much social to worry about [risas]. We try to make it as technical as possible, because I guess it tends to work better for all if we do.

Sin embargo, eso no significa que la solución pase sólo por implementar medidas técnicas, y no por convencer a la gente de que haga determinadas cosas para que esa solución funcione. No supone que no se tenga que movilizar a personas tanto como a cambios y dispositivos tecnológicos. Toda forma de acción social supone una determinada composición y articulación entre los diferentes agentes implicados.

2.2.5 *Bashims*.

Pondré un breve ejemplo.[22] Un Desarrollador muy joven me contaba en una entrevista que en Debian se dedicaba principalmente a cuestiones de Control de Calidad, *Quality Assurance*. En la página «Debian Quality Assurance» (Qa.debian.org, s.f.) se puede leer: «The Debian Quality Assurance (QA) Team tries to improve the distribution as a whole, not only a specific set of packages. It also serves as a central place for discussion about distribution-wide problems and improvements regarding quality». Y sobre todo, se dedicaba a corregir *bashisms*. En el momento de la entrevista se estaba completando el cambio de la *shell* (véase el cuadro 2.8, *Shell*) de sistema (`/bin/sh`) predeterminada en Debian, desde `bash` a `dash`. Las ventajas de este proceso eran ampliamente aceptadas, y básicamente consistían en una mejora en la velocidad y una disminución de la memoria usada en la ejecución de los *scripts* que invocaban `/bin/sh`. En principio, un mismo *script* que declare que usa `/bin/sh` como intérprete (indicándolo así al principio

[22] Al respecto se puede consultar Sobol (2009) y los enlaces que aparecen allí.

del mismo) puede ser ejecutado tanto por la *shell* `bash` como por `dash` (cuál sea la efectivamente utilizada dependerá de a cuál de ellas enlace el archivo `/bin/sh`). A no ser, precisamente, que contenga lo que se conoce como *bashisms* («bashismos»), expresiones que hacen uso de capacidades de `bash` que no están presentes en `dash`. Si los tiene, un *script* que funcionaba cuando `/bin/sh` redirigía a `bash` dejará de funcionar, o no lo hará de manera completa, cuando redirija a `dash`.

Existen dos maneras de solucionar este problema. O bien se eliminan los *bashisms*, o bien se declara explícitamente en el *script* que se debe hacer uso de `bash` para su ejecución. En ambos casos, para un Proyecto tan grande como Debian el problema consiste en la gran cantidad de paquetes de *software* que usan estos *scripts* y que es necesario cambiar. En cualquier caso, había consenso en que era un error *técnico* que un *script* con *bashisms* no lo declarara:

> **raphael:** Se tiene ya un apoyo en general por parte del Proyecto. Entonces no hay una barrera en ese sentido, sino que los *bashisms* son un error realmente y creo que actualmente nadie ha dicho que no lo son. O sea, porque técnicamente es un error hacer eso.

Así pues, había un amplio acuerdo en que era bueno cambiar la *shell* de sistema predeterminada. Quedaba el trabajo de cambiar los diferentes *scripts* que la usaban. La labor de este Desarrollador consistió precisamente en contribuir a automatizar la detección de los paquetes que padecían este problema, pero solucionarlo en cada uno de ellos era tarea, en principio, del mantenedor de cada paquete (o de otro desarrollador mediante un *Non Maintainer Upload*; véase el cuadro 3.4, NMU; en cualquier caso, un Desarrollador debía hacerse cargo de cada cambio), a los que se avisaba con un informe de fallo en el BTS:

> **raphael:** A la fecha se han corregido casi 800 paquetes, bueno se enviaron cerca de 800 reportes que ya están corregidos, y hay casi 200 más que hay que corregir. El análisis, encontrar los *bashisms*, se puede automatizar, tuvimos que mejorar las herramientas que teníamos para hacer esas búsquedas, agregarles unas pruebas. [...] siendo un sistema

Una *Shell* es una *interface* entre el usuario y el sistema operativo, aunque se suele restringir a *interfaces* de texto, intérpretes de líneas de comandos, excluyendo las interfaces gráficas.

Básicamente se pueden usar de dos formas: de manera interactiva por parte del usuario, o de manera no interactiva por parte de un *script* (un pequeño programa escrito en el lenguaje de la *shell*). En Debian, la *shell* interactiva predeterminada para los usuarios es `bash`, la segunda viene determinada por `/bin/sh`, un enlace a una *shell* específica (antes `bash`, ahora `dash`). `bash` puede hacer más cosas y tiene un conjunto más amplio de características, a costa de ser más lenta. Es mucho más útil al ser utilizada directamente por un usuario, pero más lenta en la ejecución de *scripts*.

Cuadro 2.8: *Shell*

operativo libre hay de todo, hubo muchos casos raros. Esa parte se pudo automatizar, pero para corregirlos se tiene que buscar la solución en cada caso.

Yo lo puedo corregir subiendo un paquete. Se lleva más de dos años con esto. En realidad muchos más, pero dos años empujando más el cambio. Se han subido paquetes corrigiendo estos fallos, pero en general los mantenedores mismos lo corrigen.

Cuando se suben estos paquetes, se suben a una cola de espera. Se quedan ahí unos días. El mantenedor tiene la posibilidad de cancelar ese paquete. Se le mandan los cambios al mantenedor y se le avisa, si desea que se le de más tiempo o que se cancele.

En definitiva, implementar ese cambio, ampliamente concebido como técnicamente correcto y deseable, implicó sobre todo realizar cambios en los paquetes mantenidos, pero esto sólo se pudo hacer movilizando a una gran cantidad de agentes humanos. Lo mismo que vimos que ocurrió en la disputa sobre `python` y sobre el *firmware* no libre.

2.2.6 De las cajas negras a los parches como metáfora de lo técnico.

Entonces, definir un problema como técnico implicaría, para los miembros de Debian, que su solución se simplifica. Al menos de dos formas. Por un lado, es más fácil presentar una solución (concebida como técnica), y a uno mismo si se es quien puede implementar esa medida, como punto de paso obligatorio (Callon, 1984, 205 ss.), en el sentido que vimos en la sección 2.1.7, *Procesos de traducción*. Movilizar los recursos y los agentes necesarios se hace más sencillo.

Por otra parte, permitiría aislarlo momentáneamente de su relación con otros elementos y concentrarse en un conjunto más pequeño de agentes y relaciones. Es decir, lo técnico haría referencia a lo que se encuentra, o se puede introducir, en una *caja negra*. Así explica Latour el significado de «encerrar en una caja negra»: «Cuando una máquina funciona eficazmente, cuando se deja sentado un hecho cualquiera, basta con fijarse únicamente en los datos de entrada y los de salida, es decir, no hace falta fijarse en la complejidad interna del aparato o del hecho» (Latour, 2001, pág. 262).

En esa misma obra encontramos un fragmento que parece especialmente escrito para describir la relación entre la concepción de lo técnico en Debian y la metáfora de la caja negra (la tarea principal sería aquí producir un sistema operativo lo más libre y universal posible):

> Técnico se refiere, en primer lugar, a un subprograma, o a una serie de subprogramas anidados unos dentro de otros, al estilo de las muñecas rusas, como los que hemos mencionado más arriba. Cuando decimos «ésta es una cuestión técnica», queremos decir que debemos *apartarnos* un momento de la tarea principal y que, una vez resuelta la cuestión técnica, podremos *retomar* nuestra tarea inicial, que es lo único que merece concentrar nuestra atención. Una caja negra se abre momentáneamente, aunque pronto se cerrará de nuevo, convirtiéndose en algo completamente invisible en la secuencia principal de la acción (Latour, 2001, pág. 228).

Con una salvedad. En el caso del *software* libre, las cajas negras no están cerradas (sobre todo las que se hacen con código), puesto que uno de los elementos centrales que lo definen es el código *abierto*. Una de las críticas recurrentes al *software* propietario es su carácter de caja negra que no se puede abrir. En el *software* libre pueden cerrarse para que el usuario, o cualquiera que las use, no tenga que preocuparse de los detalles, pero es trivial abrirlas.

Abrirlas no sólo para observarlas, y no sólo para modificar su funcionamiento interno de tal modo que sus entradas y salidas sigan siendo las mismas, sino para poder extenderlas, para establecer nuevas conexiones. Esto no es un añadido, es un aspecto esencial de lo que constituye un paquete de *software* libre. Como decía un Desarrollador en una entrevista:

> **mooch:** [Hablando sobre las licencias, y sobre si son un aspecto social o técnico de los paquetes] Es técnicamente aceptable porque según esa licencia tú puedes trabajar la parte técnica del paquete, o sea tú puedes modificarlo o puedes [...] el tema no es que técnicamente pueda funcionar, es que tú puedas extender la funcionalidad del paquete, o sea, [...] las cuatro libertades, el poder recibir el paquete, poder modificarlo, poder usarlo para lo que quieras y poder redistribuirlo. Si no tienes eso, estás técnicamente restringido. O sea es una cuestión social, pero que tiene una repercusión en tus posibilidades técnicas, que es muy importante. Tan importante como que te permite o te impide trabajar con ese paquete.
>
> Entonces por mucho que técnicamente sea válido un paquete, si el paquete no viene ..., y no podemos modificarlo a nuestro gusto porque tiene un *bug*, o porque tiene funcionalidad que se puede expandir, o funcionalidad que es incorrecta, entonces eso nos restringe técnicamente. [...] Esas restricciones que se han añadido ahí son puramente basadas en aspectos técnicos, que tienen una repercusión social, pero son, yo creo que son aspectos técnicos.

Pero es en una entrevista con un Desarrollador de Debian donde nos proponen un ejemplo de lo que es técnico que funciona incluso mejor que la caja negra

Un parche es un archivo que contiene modificaciones a realizar sobre otro archivo, normalmente un programa de *software*, o datos. Esas modificaciones pueden consistir en una corrección de errores, mejoras, adaptaciones, o nuevas funcionalidades. El de parche es un concepto genérico (como tal no se debe confundir por ejemplo con el programa de UNIX `patch`, una implementación específica que permite crear y aplicar parches), sin una definición precisa, y existen de una gran cantidad de tipos. Por ejemplo, en la entrada *patch* de *The Jargon File* (Raymond y Steele, 2004) se define más estrechamente como «A temporary addition to a piece of code, usually as a quick-and-dirty remedy to an existing bug or misfeature». En realidad, todo desarrollo de *software* es en último término la producción de parches entendidos en sentido amplio.

Como concepto genérico, son una parte fundamental de los Sistemas de Control de Versiones (véase el cuadro 2.5, *Sistemas de Control de Versiones y Repositorios*, y la sección 4.8, *Repositorios de* software *libre*), que guardan los cambios incrementales como parches. Son fundamentales también para el *Bug Tracking System* de Debian (véase el Cuadro 2.3, *Bug Tracking System*), donde se pueden mandar parches que corrijan *bugs* específicos. Por último, las modificaciones que hacen los mantenedores de paquetes en Debian sobre los programas de *upstream* se guardan como parches en el paquete fuente de Debian, lo que permite distribuir el código fuente íntegro de esos programas, más las modificaciones necesarias para integrarlo en Debian, sin que ambos se confundan.

Cuadro 2.9: Parches

como metáfora, el *parche* (véase el Cuadro 2.9, *Parches*):

> **vorlon:** It is clearly technical if there is a patch in dispute. If someone has a bug report that has a patch attached to it saying “you should do this”, it’s clearly a technical dispute. But then there are a lot of other things where, you know, there is some technical element to it but the ...

Aunque en su uso más habitual el concepto de parche refiere a modificaciones pequeñas en relación al tamaño del programa, y por lo tanto se presentan

aparentemente como un asunto relativamente menor, lo cierto es que se puede extender su uso metafórico a todo el proceso de desarrollo de *software*. En el límite, empezar a escribir un programa desde cero puede conceptualizarse (o metaforizarse) como la escritura de un parche sobre un archivo vacío. La figura del parche es sobre todo importante para caracterizar el desarrollo del *software* libre y la tradición de la que proviene, la cultura y la filosofía de UNIX:[23]

> But something else happened in the year of the AT&T divestiture (1983) that would have more long-term importance for Unix. A programmer / linguist named Larry Wall quietly invented the patch(1)[24] utility. The patch program, a simple tool that applies changebars generated by diff(1) to a base file, meant that Unix developers could cooperate by passing around patch sets — incremental changes to code — rather than entire code files. This was important not only because patches are less bulky than full files, but because patches would often apply cleanly even if much of the base file had changed since the patch-sender fetched his copy. With this tool, streams of development on a common source-code base could diverge, run in parallel, and re-converge. The patch program did more than any other single tool to enable collaborative development over the Internet — a method that would revitalize Unix after 1990 (Raymond, 2003, pág. 38).

Y es importante porque esta figura del parche resume y simboliza la lógica de la cooperación y del don, pero también la de la articulación de elementos pequeños y heterogéneos con orígenes y ontologías diversas frente a la del diseño total previo. Muestra la estructura distribuida y colaborativa de la producción de *software*. Pone de manifiesto la apertura de las prácticas de producción y permite comprender la importancia del uso de los Sistemas de Control de Versiones. Todas estas cuestiones se irán tratando en las secciones siguientes. Frente al concepto de caja negra, la figura del parche además enfatiza su carácter siempre abierto y examinable públicamente. Al menos cuando, como en el caso del *software* libre,

[23] Sobre la cultura y la filosofía de Unix, véase la sección 4.1, *La Red como metáfora. Unix*.
[24] El número entre paréntesis tras éste y el siguiente programa indica la sección de las páginas del manual de Unix donde se describen. La sección 1 incluye programas ejecutables y órdenes de *shell*.

donde los parches se producen y aplican respecto del código fuente y no del binario compilado, se hace evidente la naturaleza *textual* del parche. Lo que facilita, entre otras cosas, determinar su programa de acción.

El parche aparece entonces como lo técnico por excelencia. Es una metáfora en la línea de la de la caja negra, puesto que señala a un objeto concreto que se hace visible o invisible, que cobra relevancia, según aparezcan o no problemas. E igualmente se puede volver a él para examinarlo cuando es necesario. Permite examinar las conexiones que encapsula, el programa de acción que está inserto y que representa. Pero es también más interesante en cuanto que es más específica, y más cercana a la práctica concreta de los miembros de Debian: una parte importante del proceso de convertirse de Mantenedor y de ser un Desarrollador consiste precisamente en producir, aplicar y mantener parches. Junto con las bibliotecas de *software* o las A.P.I.s (*Application Programming Interfaces*), por ejemplo, son dispositivos que encarnan de manara clara las ideas de mediación y delegación de la acción de un agente. Una práctica efectiva en este campo es en numerosas ocasiones cuestión de producir un parche en este sentido genérico, producir código que produzca efectos.[25] Son también, al menos en el caso del *software* libre, fáciles de abrir y modificar.

2.2.7 Sentidos de lo social y lo técnico.

Por otra parte, lo que une a los miembros de Debian como comunidad y da sentido a sus prácticas tecnológicas, según aparece en las entrevistas, es un objetivo compartido que sitúan en el campo de lo social: producir y distribuir *software* libre. Simultáneamente, como hemos visto, definen su identidad como técnicos e ingenieros. Desde una perspectiva muy general y a largo plazo, esto parece ser una de las claves para entender Debian y el *software* libre en general: la centralidad del proceso general de encontrar formas de mediación técnica y legal (*parches* en el sentido general que hemos visto) que traduzcan, reelaboren y redefinan toda una serie de fines éticos y políticos. Como hemos visto, producir código para retomar lo verdaderamente importante.

En ese sentido, tal y como los miembros de Debian crean estos dos ámbitos,

[25] Es famosa la cita de Linus Torvalds, el creador del *kernel* Linux, en este sentido: «Talk is cheap. Show me the code» (Lkml.org, 2000).

lo social engloba lo técnico en tanto que entienden que su actividad técnica sólo tiene sentido en ese marco social. Así que tenemos una aparente paradoja. Por un lado, lo social es el resto, lo que queda tras el trabajo de mediación tecnológica. Pero también es el marco en el que se define lo técnico y las posibilidades de esa mediación. Un marco que se crea simultáneamente a partir de su contenido y en relación a él. Y es que, como seguiremos viendo a lo largo de todo este trabajo, estamos frente a procesos de constitución de un *colectivo sociotécnico* en el que son inseparables la constitución mutua de diferentes agentes sociales, humanos y no humanos. Del mismo modo, en este proceso de constitución son inseparables la construcción de convenciones y acuerdos normativos, y la realización de prácticas tecnológicas: aquéllos dan forma y son originados por éstas.

No hay que confundir aquí dos posibles significados del término «social». Por un lado tenemos lo social como constructo *emic* de los miembros de Debian, cuyo proceso de producción intentamos seguir. A este nivel pertenecen *los dos* sentidos que aparecen en el párrafo anterior: lo social como marco de la acción técnica es también parte de esa construcción *emic*. Por el otro, tenemos lo social como la producción de relaciones y vinculaciones concretas entre agentes de todo tipo, como conjunto de conexiones empíricamente trazables. Esta concepción se puede encontrar en Latour (2001, 2005):

> there is nothing specific to social order; that there is no social dimension of any sort, no 'social context', no distinct domain of reality to which the label 'social' or 'society' could be attributed; [...] 'society', far from being the context 'in which' everything is framed, should rather be construed as one of the many connecting elements circulating tiny conduits.
>
> [...]
>
> In the alternative view, 'social' is not some glue that could fix everything including what the other glues cannot fix; it is *what* is glued together by many *other* types of connectors. [...] consider social aggregates as what should be explained by the specific *associations* provided by economics, linguistics, psychology, law, management, etc. (Latour, 2005, págs. 4-5).

También en Díaz de Rada (2010):

> La referencia empírica más básica del concepto de sociedad es, complementariamente, la *vinculación concreta* entre agentes concretos. Es decir, que el concepto de «sociedad», si es que se usa analíticamente para decir algo con sentido, debe entenderse como el *proceso* de formación de vínculos sociales concretos [...] Desde esta óptica, la sociedad (socialidad) es un proceso particular, aunque fundamental de la acción humana: el proceso que consiste en formar vínculos sociales (Díaz de Rada, 2010, pág. 230).

Estamos aquí ante una categoría *etic* que no coincide exactamente con la categoría *emic* que hemos visto. Pero que tiene al menos un punto de contacto. Y es que efectivamente proporcionar el contexto de esas actividades de mediación técnica de las que venimos hablando requiere reconstruir esas conexiones y vinculaciones concretas entre realidades y agentes heterogéneos. Sólo situándolas ahí es posible entenderlas. Pero a diferencia de la categoría *emic*, ese contexto no se entiende como un distinto ámbito de realidad, sino como el conjunto de relaciones y conexiones producidas entre todos los elementos implicados. Desde esta perspectiva, no hay diferencia de principio entre las actividades que los miembros de Debian llaman técnicas y las que llaman sociales: tan productora de lo social (en sentido *etic*) es el trabajo de enviar un parche al BTS como discutir qué se debe entender por *libre* en la expresión «*software* libre», por poner un ejemplo. Pero podemos observar cómo el establecimiento de mediaciones que llaman técnicas hace aparecer simultáneamente, en una suerte de negativo, el ámbito de lo social que, para ellos, a la vez da sentido y queda fuera de esa concreta labor de mediación.

Algo similar ocurre con el término «técnico». Junto a la concepción *emic* de lo técnico que venimos considerando, encontramos la concepción *etic* de Latour (2005) que lo entiende en términos de subprogramas de acción, de delegación o de cajas negras (véase la sección 2.2.6, *De las cajas negras a los parches como metáfora de lo técnico*). Se podría entender la diferencia entre lo técnico y lo social

que hemos encontrado en Debian a partir de la diferencia que establece Latour entre programa de acción y antiprograma:

> **programas de acción, antiprogramas**. Son términos propios de la sociología y la tecnología que se han venido usando para conferir a los artefactos su carácter activo y a menudo polémico. Cada uno de los mecanismos anticipa lo que los demás actores, tanto humanos como no humanos, puede hacer (programas de acción), aunque puede que esas acciones anticipadas no tengan lugar debido a que los otros actores tengan diferentes programas, es decir, antiprogramas desde el punto de vista del primer actor. De ahí que el artefacto se encuentre en la primera línea de una controversia entre los programas y los antiprogramas (Latour, 2001, Glosario, p. 368).

En este sentido, se podría uno concentrar en lo técnico cuando está claro cuál es el programa de acción que se quiere inscribir en un dispositivo, entendiéndose lo social también, en ocasiones, como la necesidad de tratar con los antiprogramas de los demás.

Como no podía ser de otra manera dada la condición de expertos en la producción y distribución de tecnología de los miembros de Debian, estas dos concepciones (*emic* y *etic*) de lo técnico son extraordinariamente cercanas. Pero no equivalentes, puesto que un programa de acción puede ser inscrito en un documento como la Constitución de Debian, o la definición de lo que es *software libre* «caja-negrizada» en las DFSG, documentos que los miembros de Debian no llamarían técnicos. En el presente trabajo quedará suficientemente claro en cada contexto qué noción se está manejando. En cualquier caso, la utilización de términos como traducción o mediación permite eludir esta dificultad.

Lo técnico y lo social aparecen como realidades en proceso, como referentes del discurso y de las prácticas en constitución, esencialmente ligados a las acciones sociales que les dan forma. Y por eso no es de extrañar que sean ambiguos en el discurso de los implicados, que no estén estabilizados. Son formas convencionales de la acción social, que se forman a partir de las prácticas sociales concretas,

y que sólo se pueden entender en relación a ellas. No es tanto que lo social sea el marco convencional en el que se produce lo técnico, como que es la propia distinción lo que es convencional. La última cita que veíamos de Díaz de Rada (2010) seguía: «así como la cultura es una propiedad de esa acción: su forma convencional». En Debian lo que encontramos es una vinculación concreta entre agentes que son humanos, a través de, y con, dispositivos tecnológicos. Y la forma que tienen esas vinculaciones es convencional. Es una forma que surge de la interacción, en procesos de comunicación social. En la configuración de esa forma convencional es fundamental el papel que juegan los propios dispositivos tecnológicos. Por ejemplo la convención que consiste en citar, en un mensaje de correo, la parte del mensaje a la que se está respondiendo. Sin duda la convención sería diferente si los clientes de correo no estuvieran configurados para hacerlo por defecto. O el hecho de que cada *commit* en `git` lleve un comentario (que el programa pide por defecto; es posible dejarlo en blanco, pero no que no exista). O que al mandar un *bug* se mande determinada información que recoge `reportbug` (una herramienta para facilitar el envío de informes de fallo). Cuando se reporta un *bug*, se establece una relación con el mantenedor del paquete. Pero también con el registro de ese *bug* en el BTS y la discusión posterior. O sobre todo, las relaciones que se establecen entre los Desarrolladores y Mantenedores de Debian a través de sus paquetes y de la infraestructura del Proyecto. Los dispositivos técnicos cambian la forma de los actores y sus puntos y modos de articulación, por lo que transforman también sus relaciones.

Por eso lo que describimos es un colectivo o trama sociotécnica, en la que los vínculos con no humanos son esenciales. Seguiremos desarrollando este concepto en el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*

2.2.8 Agencia y semiótica.

Recurrir ahora a la teoría de la agencia de Kockelman (2007) nos permitirá introducir alguna precisión. Kockelman parte de la teoría semiótica de Peirce para conceptualizar la agencia. Distingue entre agencia residencial y agencia representacional. La primera sería

> the degree to which one can (1) control the expression of a sign (e.g., determine where and when it may be expressed), (2) compose a sign-object relation (e.g., determine what object a sign stands for and/or which sign stands for that object), and (3) commit to an interpretant of this sign-object relation (e.g., determine what effect the expression of the sign will have so far as it stands for that object) (Kockelman, 2007, pág. 376).

La segunda, por el contrario, se refiere al

> degree to which one can (1) thematize a process (e.g., determine what we talk about), (2) characterize a feature of this theme (e.g., determine what we say regarding what we talk about), and (3) reason with this theme-character relation (e.g., determine what we conclude from, or use to conclude, what we say regarding what we talk about) (Kockelman, 2007, pág. 376).

La primera tiene que ver con el poder y la capacidad de elección, la segunda con el conocimiento y la conciencia.

Voy a intentar una lectura de los procesos presentados (la disputa sobre el mantenimiento de `python` y las polémicas sobre la inclusión de *firmware* no libre) como proceso semiótico según el modelo de Kockelman. Como en todo proceso semiótico, al menos según Peirce, tenemos que distinguir tres componentes: signos (lo que está por otra cosa), objetos (aquello por lo que está el signo) e interpretantes (lo que el signo crea en tanto está por un objeto) (Kockelman, 2007, pág. 376). Lo que faltaría en el esquema de Kockelman es el papel del intérprete. Así que corregiremos la definición de interpretante en línea con Díaz de Rada (2013): lo que el intérprete del signo hace al tomar al signo por representación de un objeto. Se enfatiza así el papel del intérprete y de la acción en la producción de significado. En las primeras fases de mi trabajo de campo, y en el trabajo de investigación que precedió a esta tesis doctoral, entendía que debía tratar «técnico» y «social» como dos signos, y determinar cuál era su significado, qué era lo que representaban. Pero se perdía entonces el carácter esencialmente procesual de todo proceso

de significación. Mucho más productivo e iluminador parece, por el contrario, tomarlos como objetos según la clasificación de Peirce, lo que permite comprender cómo son producidos. Pues un objeto en este modelo de significación no es más que aquello a lo que los diferentes interpretantes apuntan:

> Objects, then, are relatively abstract entities by definition. They should not be confused with "objects" in the Cartesian sense of *res extensa*, nor should they be confused with the "things" that words seem to stand for. Indeed, it is best to think of the object as a *correspondence-preserving projection* from all interpretants of a sign. It may be more or less precise and more or less consistent (Kockelman, 2007, pág. 378).

Los objetos no son pues algo preexistente, sino que se constituyen como objetos solamente en relación al signo, en el proceso social de significación.

Tomemos el caso de `python`. Obviamente, la controversia presentada es mucho más que un proceso semiótico, y desde luego en cuanto proceso semiótico no se agota en los elementos que voy a discutir. Pero a partir del envío del informe de fallo, lo que entre otras muchas cosas hacen los participantes en la discusión es tomarlo como signo, y ponerlo en relación con un objeto. En este caso, y entre otros, los objetos serían el campo de lo técnico y lo social. Puede ponerse en relación con dos objetos diferentes, y con su propia diferencia, porque es un signo complejo, compuesto él mismo por una diversidad de signos, y que a lo largo de la controversia induce la producción de otros signos. Pero lo relevante son los interpretantes que van produciendo los agentes, y que constituyen al signo como tal. Si el informe de fallo, por ejemplo, es un signo, es sólo porque los participantes en la discusión proponen interpretantes sobre su relación con el objeto. Díaz de Rada (2013) lo expresa así a propósito de la teoría de Kockelman:

> El intérprete de hecho forma el signo cuando, al tomar una cosa como representación de otra, convierte a la primera en signo de la segunda (el objeto), produciendo un interpretante para esa relación de representación.

Kockelman (2007, pág. 378), siguiendo a Peirce, distingue tres tipos de interpretantes: afectivos, energéticos y representacionales. Dadas las características de la comunicación electrónica, en nuestro caso es difícil y forzado distinguirlos. Pero sí podemos indicar algunos, tal y como aparecieron en la descripción: señalar que el informe de fallo responde a desavenencias personales, recordar las dificultades en la transición a `python2.6`, insistir en la necesidad de definir con claridad el desarrollo de una política de empaquetado para `python`, sostener que el mantenimiento no se debe realizar en solitario, inscribir el nombre de más mantenedores en el paquete fuente, desarrollar un nuevo *helper* como respuesta a uno de los problemas planteados, son todas ellas acciones y proposiciones que, entre otras cosas, funcionan como interpretantes de la relación entre el signo o el conjunto de signos (el informe de fallo y los mensajes y acciones subsiguientes) y el objeto, la diferencia entre lo técnico y lo social. Son formas de determinar si estamos ante una cuestión técnica o social, y por lo tanto quién está legitimado para resolverla y de qué manera. Algunos de estos interpretantes se convierten a su vez en nuevos signos, como por ejemplo el nuevo *helper*, que mantiene su propia relación con el objeto (lo técnico) a través del interpretante que es su acción en el *software* empaquetado con su ayuda.

Algo similar ocurre en la controversia sobre el *firmware* no libre. En los conjuntos de acción relativos a esta controversia, una votación sobre la publicación del sistema operativo con *blobs* binarios no libres, o la efectiva separación de estas partes en un momento posterior, se constituyen como signos cuando se compone una relación con el objeto mediante una serie de interpretantes. «Violar las DFSG va en contra de nuestro compromiso con la libertad del *software*», «pero no hacerlo viola nuestro compromiso con los usuarios», «necesitamos una Resolución General para resolver esta cuestión», o «publicamos por fin una versión del sistema operativo sin *firmware* no libre», son interpretantes que componen una determinada relación entre los signos que aparecen en la controversia y los objetos en producción que son lo técnico y lo social.

¿Qué aporta este modelo semiótico a la comprensión etnográfica de las formas de vida social que se producen en el Proyecto Debian? Fundamentalmente dos cosas. La primera es que permite comprender que los ámbitos de lo técnico y lo social son una realidad en proceso, objetos que se están creando, que se están

incoando a partir de esa serie de interpretantes. Es un proceso semiótico, pero al mismo tiempo un proceso ontológico que ayuda a dar cuenta de la producción y construcción de estos dos ámbitos.

Pero sobre todo, en segundo lugar, porque ofrece paralelamente una teorización de la agencia en términos semióticos, conceptualizándola como el grado de control que se tiene sobre los diferentes elementos de ese proceso semiótico, un control que es multidimensional, graduado y distribuido. Tomemos la agencia residencial. Como hemos visto, es multidimensional y graduada porque los agentes pueden tener un grado de control variable sobre tres dimensiones de la misma.[26] Así, se puede *controlar la expresión del signo* controlando cuándo y dónde aparece éste: un informe de fallo al BTS antes de la publicación de una nueva versión, el desarrollo de un nuevo *helper*, o la propuesta de una Resolución General sobre la inclusión del *firmware* no libre. O *componer la relación entre ese signo y un objeto* determinando que esos signos están refiriendo *también* al campo de lo técnico o de lo social. Por fin, *comprometerse con un interpretante* implica determinar el efecto de la expresión de ese signo en tanto está por el objeto: afirmar la necesidad de mantener un *software* en equipo o insistir en las necesidades de los usuarios, por ejemplo. Ninguno de los agentes tiene un control total sobre ninguna de estas dimensiones, sino que este control es una cuestión de grado y está distribuido entre los agentes participantes. Por ejemplo, producir el signo que es el informe de fallo depende en primer lugar de quienes lo proponen, pero que se produzca en el BTS (y que siga abierto durante todo ese tiempo, así como que aparezca también simultáneamente en la lista de correo del Comité Técnico) depende también del funcionamiento específico del mismo BTS, de que exista un pseudo-paquete para el Comité Técnico, o de que éste no lo cierre prematuramente. En el caso del *firmware* no libre, la solución sólo podía surgir de una cadena de prácticas que enlazaba a los desarrolladores del *kernel upstream*, el *Debian Kernel Team* y el *Debian CD Team*. Tampoco nadie podía imponer por sí sólo el interpretante que afirma que el *firmware* es *software*, y que por lo tanto el *firmware* debe vincularse a los requisitos sobre la libertad del *software*, o el contrario.

En ambos casos, que se componga la relación con el ámbito de los problemas técnicos o sociales depende de la interacción y negociación entre todos los

[26] Véase Kockelman (2007, 387 ss.).

implicados. La responsabilidad en cuanto a los interpretantes vistos y que juntos conforman el objeto, depende igualmente, como vimos, de todos los agentes implicados. La agencia es algo que claramente aparece distribuido a lo largo de la cadena de prácticas y agentes implicados en ese proceso semiótico:

> it is not usually a concrete entity –qua participant in an interaction– that determines participants' control, composition, or commitment but rather the temporally unfolding interaction itself. In other words, the locus of agency may often rest not in the individuals but rather in their ongoing interactions and the institutions that enable these (Kockelman, 2007, pág. 382).

En cuanto a la agencia representacional, depende de la capacidad para caracterizar, describir y razonar en torno a estos procesos. Se hace presente en el nivel del discurso, pero está en continuidad con la agencia residencial. En los casos vistos, no siempre es fácil separarla de la residencial, puesto que producir un signo o un interpretante implica en numerosas ocasiones ser capaz de tematizar el proceso y razonar sobre él. En el ejemplo sobre el mantenimiento de `python`, aparece cuando es necesario tematizar discursivamente dónde está lo técnico del asunto, mientras que acciones como desarrollar los *helpers* o separar del *kernel* las partes no libres y moverlas a la sección *non-free* del archivo corresponderían a la agencia residencial. La agencia representacional también aparece como un elemento fundamental, en tanto que la caracterización de un proceso o una discusión como técnico, social o una mezcla de ambos tiene consecuencias respecto a quién tiene agencia para intervenir y cómo ésta se distribuye, como seguiremos desarrollando en el capítulo siguiente. En los fragmentos de entrevistas ofrecidos se observan los desplazamientos y las inseguridades de los implicados en torno a la reflexión y razonamiento sobre esta cuestión.

Los dispositivos mencionados en estas descripciones tienen entonces agencia porque contribuyen a determinar cómo y cuándo se expresan los signos, con qué objetos se compone su relación, y qué interpretantes se producen. Pero sobre todo porque intervienen decisivamente en el proceso de constitución de los agentes. Si la agencia no reside tanto en entidades individuales sino en su inter-

acción, la interacción mediante estos dispositivos tecnológicos constituye literalmente agentes nuevos, agentes que tienen la capacidad de hacer cosas diferentes que los individuos considerados en solitario. Estos dos sentidos no son independientes. A fin de cuentas, la agencia residencial en el sentido de Kockelman sólo tiene sentido predicada de los agentes en relación, agentes que son ellos mismos constituidos por sus relaciones.

Esta concepción semiótica de la agencia es consistente con la ofrecida por Latour. En *The Pasteurization of France* dice Latour (1988, pág. 35):

> I am not using the word "agent" in any metaphorical or ironical sense but in the semiotic sense. Indeed, the social link is made up, according to the Pasteurians, of those who bring men together and those who bring the microbes together. We cannot form society with the social alone. We have to add the action of microbes.

Aunque Latour recurre a otra tradición semiótica, notablemente la de Greimas, existen puntos en común,[27] como la estrecha relación de los procesos de significación con la agencia, la diversidad de procesos semióticos y la importancia de los objetos materiales en ellos, o la concepción de la semiosis a partir de la relación entre elementos heterogéneos (proposiciones, inscripciones, acciones, objetos materiales, ...). Quizás la más importante sea la materialidad con la que se produce la significación. Para Latour, la referencia no designa una relación externa entre un signo y una realidad ya dada, sino la serie de transformaciones e inscripciones que trasladan y traducen signos y objetos (Latour, 2001, cap. 2). Determinar la expresión y el significado de un signo supondría determinar en qué cadenas de transformación se inscribe, como las que venimos analizando. Cabría pensar que lo que Latour llama proposiciones es algo muy parecido a los interpretantes de Kockelman:

> No utilizo este término en el sentido epistemológico de una oración que se juzga verdadera o falsa (para esto prefiero reservar la palabra

[27] Para una discusión de la relación de Latour y la Teoría del Actor-Red con la semiótica pueden verse Høstaker (2005) y Law (2008).

> «enunciado»), sino en el sentido ontológico de lo que un actor ofrece a otros (Latour, 2001, Glosario, p. 368).
>
> Las proposiciones no son afirmaciones, ni cosas, ni ningún tipo de intermediario entre las dos. Son sobre todo actantes. [...] son *ocasiones* que las distintas entidades tienen para establecer contacto [...] La relación que se establece entre las distintas proposiciones no es una relación de correspondencia a uno y otro lado de una inmensa brecha, sino lo que denominaré *articulación* (Latour, 2001, pág. 169).

Esto es lo que han estado haciendo nuestros agentes en los procesos analizados: articular proposiciones. En fin, insistiré solamente en dos aspectos de la semiótica tal y como es utilizada por Latour. Tal como él mismo escribe: «Semiotics provides me with a discipline, [...] the semiotic method is here limited to the interdefinition of actors and to the chains of translations» (Latour, 1988, pág. 11).

Que ambos aspectos sean abordables desde una perspectiva semiótica no es independiente. Poco antes, Callon y Latour escribían:

> By translation we understand all the negotiations, intrigues, calculations, acts of persuasion and violence, thanks to which an actor or force takes, or causes to be be conferred on itself, authority to speak or act on behalf of anoher actor or force (Callon y Latour, 1981, pág. 279).

Y en ese punto introducen una nota:

> By the term 'actor' we mean, from now on, the semiotic definition by A. Greimas in *Dictionnaire de semiotique* (Paris: Hachette, 1979): 'whatever unit of discourse is invested of a role', like the notion of force, it is no way limited to 'human' (Callon y Latour, 1981, 301, n.8).

En fin, una concepción semiótica de la agencia define a un agente o actor más a partir de sus efectos que de su intencionalidad, de sus efectos en la producción

de significado y en la constitución de objetos, incluyendo en nuestro caso qué son lo técnico y lo social. Y siempre en relación a la constitución de otros agentes y la definición de sus posibilidades de acción:

> an actor says what I want, what I know, what I can do, marks out what is possible and what impossible, what is social and what technical (Callon y Latour, 1981, pág. 288).

Así, los procesos de traducción y de constitución de agentes aparecen como procesos semióticos. El punto fundamental es que el agente se define por su lugar en una cadena de prácticas, de traducciones, en relación a otros actores. Determinar la expresión y el significado de un signo es determinar en qué cadenas se inscribe (Latour, 2001). Como hemos visto y seguiremos viendo a lo largo del presente trabajo, entender la agencia como la capacidad de controlar qué se expresa es también determinar qué lugar ocupa un actor en un colectivo.

3 Formas de gobierno. Doocracia, agencia y apertura.

> *Siempre se ha dicho que todos somos voluntarios y que nadie puede obligar a nadie a hacer algo, porque aquí no hay una jerarquía de jefes ni nada. O sea, los equipos con poder que hay en Debian, los equipos que tienen privilegios en Debian, es para hacer tareas, pero no para mandar.*
>
> — Dato, Desarrollador de Debian, Alicante, mayo de 2009.

En realidad, la cuestión enunciada en el título de esta sección recorre todo el presente trabajo, sin que sea posible aislarla y tratarla separadamente de los procesos sociales descritos a lo largo de los diferentes capítulos. Además, en el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*, se profundizará en la cuestión de la estructura y la historia del Proyecto Debian.

Sin embargo, conviene detenerse ahora en dos aspectos fundamentales: la descripción de la doocracia como forma de gobierno y de toma de decisiones, y la constitución de Debian como un «público recursivo». Estos dos aspectos tienen en común que nos llevan a considerar la cuestión de la organización y el gobierno de manera inseparable de la relación entre humanos y dispositivos tecnológicos. Además, nos detendremos en la relación de estos aspectos con dos elementos que ya hemos considerado, y que seguirán apareciendo: la agencia y su distribución,

y la centralidad de la apertura. Antes de todo ello se ofrece una breve descripción de la estructura formal del Proyecto Debian y de su historia.

3.1 Historia y organización del Proyecto Debian.

La historia de Debian ha sido contada en numerosas ocasiones.[1] Aquí sólo se ofrece un breve resumen de algunos de los momentos importantes de esta historia. El Proyecto Debian fue fundado en 1993 por Ian Murdock, entonces un estudiante de Ciencias de la Computación en la Universidad de Purdue (Indiana, EE.UU). Ese mismo año escribe el «Debian Manifesto» (apéndice C, *The Debian Manifesto*), en el que explica sus motivos e intenciones para crear Debian. De lo que se trataba era de crear una distribución de Linux (véase el Cuadro 1.7, *Distribuciones GNU/Linux*), un concepto por entonces relativamente nuevo. Desde el principio, se pretende conseguir una distribución de alta calidad, no comercial, construida y mantenida colaborativamente por sus usuarios según el modelo de Linux y de GNU, y según las necesidades de los usuarios y no los de alguna compañía. Estos siguen siendo hoy principios fundamentales de Debian. Así, el Proyecto Debian es una organización voluntaria que mantiene una *Community-driven distribution*, una «distribución mantenida por la comunidad», independientemente de cualquier compañía o empresa.[2]

Técnicamente, lo que ha distinguido a Debian como distribución ha sido el desarrollo de herramientas de manejo de paquetes (véase la sección 4.5, *El empaquetado como práctica sociotécnica*) de alta calidad que simplificaban mucho la instalación de *software*, y la progresiva ampliación a diferentes arquitecturas, 10 en la última versión (Debian.org, s.f.[l]).

En el año 1997, a iniciativa del segundo líder del Proyecto Bruce Perens, se

[1] Se puede consultar por ejemplo Krafft (2005, cap. 2), Sadowski *et al.* (2008), la página de historia en la *web* del Proyecto (Debian.org/doc, s.f.[c]), el «Timeline of the Debian Project» (Timeline.debian.net, s.f.), y sobre todo Coleman (2013, cap. 4). Un documento muy interesante es la discusión sobre la historia de Debian organizada por Gabriella Coleman en DebConf 4, en Coleman (s.f.).

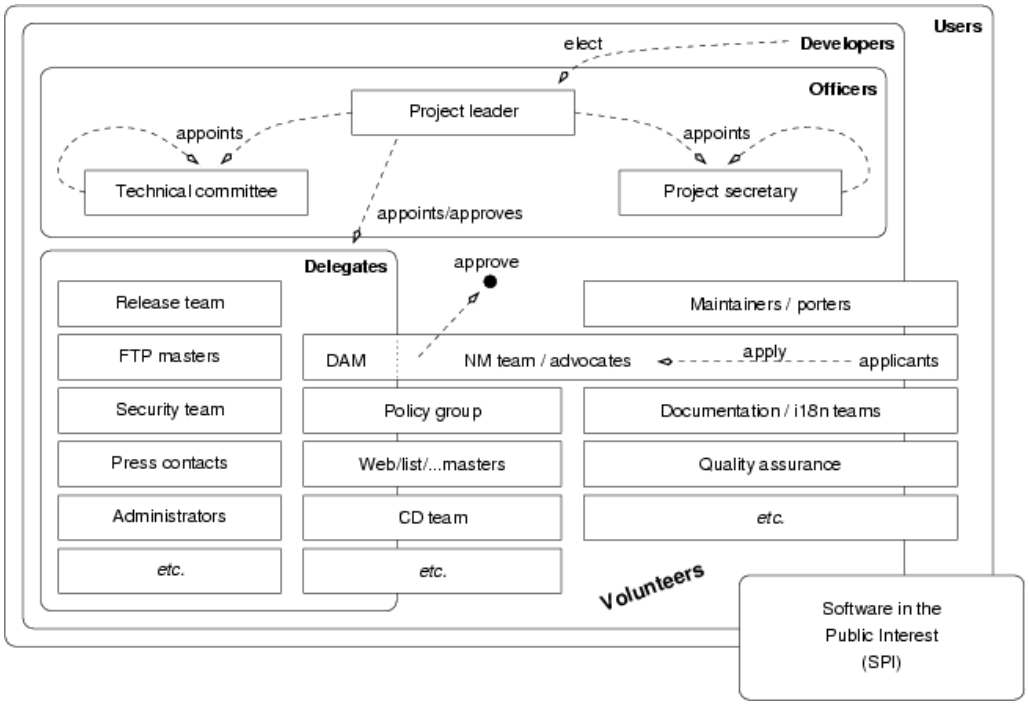
[2] A diferencia de distribuciones como Red Hat o SUSE Linux. Sobre este concepto se profundizará más adelante. Véase la sección 6.1.1, *Los sentidos de la comunidad*.

publican los dos documentos más importantes que definen los objetivos comunes del Proyecto, el «Contrato Social de Debian» y las «Directrices de *Software* Libre de Debian» (apéndices A, *Contrato social*, y B, *DFSG*), y en 1998 la Constitución de Debian (Debian.org, s.f.[i]), que describe la estructura organizacional de Debian y establece los mecanismos para la toma de decisiones. Los cuatro puntos que componen el Contrato Social son: «Debian permanecerá 100% libre» (según las DFSG), «Contribuiremos a la comunidad de *software* libre» (usando licencias libres al escribir nuevo *software* y colaborando con los autores originales, *upstream*), «No ocultaremos los problemas» (los *bugs* del BTS son públicos), y «Nuestra prioridad son nuestros usuarios y el *software* libre» (y no intereses comerciales). A esto se añade una indicación de que los programas que no cumplan las DFSG podrán incluirse en otras áreas del archivo que no forman propiamente parte de Debian (véase la sección 5.2.3, *Las licencias en Debian*).

Durante ese periodo se produce lo que Coleman denomina «Debian's historical transition from an informal group (organized largely around charismatic leadership, personal relationships, and ad hoc decision making) to a stable institution» (Coleman, 2013, pág. 125). Así, en 1999 empieza a elegirse formalmente por votación de todos los miembros del Proyecto al *Debian Project Leader*, y se formaliza el proceso para convertirse en miembro de Debian, el *New Maintainer Process* (véase la sección 6.3.1, *El* New Maintainer Process). Todos estos cambios pretenden asegurar la coherencia de los objetivos y procedimientos comunes en una época de rápido crecimiento en el número de miembros.

Los acontecimientos más importantes que puntúan la historia de Debian son las diferentes *releases* (publicaciones o lanzamientos) de su distribución de *software*, que marcan en gran medida el ritmo del Proyecto. A partir de 1996, las diferentes versiones de Debian usan un nombre además de un número de versión. Este nombre es siempre el de un personaje de la película *Toy Story*, dado que Bruce Perens trabajaba en ese momento en Pixar.

A lo largo de los siguientes capítulos se irán describiendo diferentes episodios de esta historia, especialmente en la sección 6.3, *Segunda controversia: ¿Quién es miembro de Debian?*



Cuadro 3.1: Esquema de la organización de Debian

En cuanto a su organización, nos limitaremos aquí a una visión muy general, puesto que se profundiza en los aspectos relevantes en otros lugares a lo largo del presente trabajo.[3] Debian es una asociación de voluntarios, en la actualidad poco más de mil.[4] Los miembros de pleno derecho del Proyecto son llamados Desarrolladores de Debian (sobre los tipos de miembros que existen, sus derechos y los cambios que se han producido, véase el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*). El cuadro 3.1, *Esquema de la organización de Debian*, elaborado por el Desarrollador de Debian Martin Krafft (Krafft, 2005, pág. 47), ofrece una visión sintética de la estructura del Proyecto. Algunos de sus elementos ya se han explicado, y otros se irán introduciendo en las secciones y capítulos siguientes.

Normalmente, la labor básica de los Desarrolladores consiste en mantener

[3] Para una visión general sobre la organización y estructura de Debian, pueden verse Krafft (2005, cap. 2) y Coleman (2013, cap. 4). Un trabajo sobre la evolución de las formas de gobierno en Debian puede encontrarse en O'Mahony y Ferraro (2007). Una página muy útil sobre la organización de Debian, detallando quién ocupa cada puesto, es Debian.org (s.f.[m]).

[4] Sobre el número exacto de miembros de Debian, véase la nota 6 de la introducción.

un número de paquetes de *software*, integrándolos en la distribución. Cada Desarrollador es responsable de los paquetes de *software* que mantiene, y de las diversas labores que realiza en el Proyecto, como mantener la infraestructura o coordinar diferentes partes de la actividad del Proyecto. Es cada vez más común que la actividad de mantenimiento de paquetes de *software*, así como las demás labores de los Desarrolladores, se realicen en el seno de diferentes equipos. Algunos de estos son, como veremos, centrales para el mantenimiento de la infraestructura y los servicios del Proyecto. No siempre se puede establecer una frontera clara entre ambos tipos de actividad, el empaquetado y las tareas generales o de mantenimiento, y la doocracia hay que entenderla referida a ambas.

A lo largo de su historia Debian ha crecido enormemente, tanto en número de Desarrolladores como en número de paquetes o programas mantenidos. El número de Desarrolladores ha crecido desde unas pocas docenas de los primeros años, a algo más de mil en la actualidad. Este número se ha mantenido estable durante los últimos años. De ahí las dificultades de coordinación del Proyecto. Y de ahí también lo asombroso del Proyecto. Como dicen Gonzalez-Barahona *et al.* (2009):

> With respect to the absolute figures, it can be noted that Debian 4.0[5] is probably one of the largest coordinated software collections in history, and almost certainly the largest one in the domain of general-purpose software for desktops and servers. This means that the human team maintaining it, which has also the peculiarity of being completely formed by volunteers, is exploring the limits of how to assemble and coordinate such a huge quantity of software. Therefore, the techniques and processes they employ to maintain a certain level of quality, a reasonable speed of updating, and a release process that delivers usable stable versions, are worth studying, and can for sure be of use in other domains which have to deal with large, complex collections of software (Gonzalez-Barahona *et al.* 2009, pág. 281).

[5] Llamada *Etch*, y publicada en 2007. La actual es la 9, llamada *Stretch* y publicada en 2017. Consiste en más de 51.000 paquetes de *software*.

La comunicación y coordinación en el Proyecto se llevan a cabo a través de las listas de correos oficiales, canales de IRC, conferencias presenciales, y toda una infraestructura de servicios que permiten la colaboración en el desarrollo. La cara pública más importante de esta infraestructura es el *Bug Tracking System* (BTS) o Sistema de seguimiento de fallos. Sobre éstos sistemas véanse las secciones 1.3, *Cuestiones de metodología*; 2.1, *Bug #573745*; 3.3, *Debian como público recursivo*; y 6.2, *Interacción entre aspectos* online *y* offline. *DebConfs*.

El cargo más visible es el de Líder del Proyecto Debian (*Debian Project Leader* o *DPL*), elegido anualmente entre los desarrolladores de Debian por votación. Su papel es representar oficialmente al Proyecto Debian, y realizar funciones de coordinación y comunicación, en gran parte a través del nombramiento de delegados. Se pueden tomar decisiones mediante una Resolución General, una votación realizada según el método Condorcet. Estas se pueden utilizar para rechazar una decisión del DPL, reformar la constitución del Proyecto, o tomar una decisión que afecte a todo el Proyecto. Sobre estos elementos y otros cargos de Debian, véase la sección 3.2, *Toma de decisiones: de la meritocracia a la doocracia*.

Por último, *Software in the Public Interest* (*SPI*) (SPI.org, s.f.) es la organización sin ánimo de lucro que posee la marca Debian y sus dominios en Internet, canaliza las donaciones al Proyecto y da cobertura legal al Proyecto Debian. Es la más importante de las *Trusted Organizations* (Wiki.debian.org, s.f.[h]), organizaciones que pueden manejar recursos en nombre de Debian. La SPI fue creada por Bruce Perens, y ahora ofrece cobertura legal también a otros proyectos de *software* libre.

3.2 Toma de decisiones: de la meritocracia a la doocracia.

Vamos a considerar ahora las formas de gobierno del Proyecto Debian. Es decir, la cuestión de cómo, y por parte de quién, se toman las decisiones. En este sentido, será fundamental determinar las lógicas complementarias pero no siempre libres de tensiones de la democracia, la meritocracia, el consenso o la doocracia como formas de gobierno. Es necesario comprender los procesos de toma de

decisiones para poder comprender el carácter distribuido de la agencia, que empezamos a considerar en la sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*. Y puesto que estamos considerando la agencia en entornos de acción social fuertemente mediados por la tecnología y el *software*, la comprensión de la mediación tecnológica de la doocracia se complementará con la consideración de la naturaleza del *software* como objeto híbrido (véase el capítulo 4, *El carácter híbrido del* software. Bugs, hackers, *paquetes y repositorios*).

Los diferentes procesos de toma de decisiones en el Proyecto Debian están formalizados en la Constitución de Debian (Debian.org, s.f.[i]), cuya versión actualmente en vigor es la 1.7, ratificada en 2016. La primera versión se aprobó en diciembre de 1998.

Según esta Constitución, las decisiones pueden ser tomadas por:

1. Los Desarrolladores, mediante una Resolución General o una elección.
2. El Líder del Proyecto.
3. El Comité Técnico o su Presidente.
4. El Desarrollador que trabaja en una tarea determinada.
5. Delegados nombrados por el Líder del Proyecto para tareas específicas.
6. El Secretario del Proyecto.

Ya hemos tratado sobre las funciones y el funcionamiento del Comité Técnico (véase el cuadro 2.2, *Algunos poderes del Comité Técnico*). En cuanto a las Resoluciones Generales, de acuerdo con la Constitución son empleadas fundamentalmente para los siguientes asuntos. El primero es elegir al Líder del Proyecto. El segundo es como recurso para tomar o anular decisiones que deben ser tomadas por otras instancias: el Líder del Proyecto o algunos de sus delegados, o el Comité Técnico; también para enmendar la propia Constitución. El tercero es la aprobación de documentos *no técnicos*; entre estos los más importantes son los que aparecen en la propia Constitución como Documentos Fundamentales (*Foundation Documents*), el «Contrato Social» de Debian (apéndice A, *Contrato social*,

y las «Directrices de *Software* Libre de Debian» (apéndice B, *DFSG*). Por último, tomar decisiones sobre los recursos que manejan las *Trusted Organizations*.

El proceso de propuesta, discusión y aprobación de una Resolución General es de hecho complejo, como se puede comprobar cualquiera de los ejemplos que aparecen en la página *web* «Debian Voting Information» (Debian.org/vote, s.f.):

> Debian uses the Condorcet method for project leader elections (the wikipedia article linked to is pretty informative). Simplistically, plain Condorcet's method can be stated like so
>
> > Consider all possible two-way races between candidates. The Condorcet winner, if there is one, is the one candidate who can beat each other candidate in a two-way race with that candidate.
>
> The problem is that in complex elections, there may well be a circular relationship in which A beats B, B beats C, and C beats A. Most of the variations on Condorcet use various means of resolving the tie. See Cloneproof Schwartz Sequential Dropping for details. Debian's variation is spelled out in the the constitution, specifically, § A.6.

Así pues, el más importante proceso de toma de decisiones se formaliza en la Constitución de Debian como una votación democrática. Sin embargo, las votaciones sobre una propuesta de Resolución General se producen raramente: desde 1998 se han producido 52 votaciones, de las cuales 19 corresponden a la elección anual del Líder del Proyecto, y 11 a enmiendas a la propia Constitución. Y sobre todo, son consideradas con una cierta desconfianza por parte de los miembros de Debian, y se intenta limitar su uso lo más posible. Desde luego, no para resolver cuestiones que sean conceptualizadas como «técnicas». Como ya hemos visto, esto es interpretado como el uso de «soluciones sociales» para resolver «problemas técnicos» en la mayoría de los casos (véase la sección 2.2.3, *La producción práctica de la distinción*), y se desaconseja fuertemente.

El Líder del Proyecto (DPL) es elegido por periodos de un año. Es el representante oficial del Proyecto Debian, y se espera de él también que ejerza un papel de mediación, coordinación y facilitación de la comunicación cuando sea necesario (Debian.org, s.f.[n]). Las atribuciones que le da la Constitución son básicamente nombrar Delegados respecto a áreas de responsabilidad específicas, dar autoridad a otros Desarrolladores, y tomar decisiones urgentes o sobre las que ningún otro tenga responsabilidad.[6] Así pues, es una figura muy diferente de la que en otros Proyectos se suele llamar, de manera humorística, Dictador Benevolente, en la que una figura de prestigio, normalmente el fundador del proyecto, mantiene para siempre la capacidad de tomar decisiones que afectan al proyecto (Wikipedia, s.f.[d]). Durante las campañas electorales anuales para elegir al DPL se especifican diferentes visiones y estrategias para el Proyecto en las plataformas electorales de los candidatos, así como en las numerosas preguntas y respuestas a los mismos en la lista `debian-vote` (Debian-vote, s.f.).

El Secretario del Proyecto tiene como atribuciones organizar y conducir las votaciones, sustituir al Líder del Proyecto en caso de necesidad, y resolver disputas sobre la interpretación de la Constitución (Debian.org, s.f.[o]).

Los Delegados del Líder del Proyecto son elegidos y reemplazados por éste, aunque el Líder no puede anular las decisiones tomadas por algún Delegado. Sus atribuciones son determinadas por el propio Líder. Así, el poder fundamental del DPL es la delegación, que puede cambiar una situación que se perciba como problemática, o legitimarla. Lo más relevante en estas figuras es que pueden tomar decisiones que quien los nombra no puede tomar por sí mismo, como medida para evitar la concentración de poder. Así lo establece la Constitución:

> Los Delegados del Líder del Proyecto:
>
> 1. disponen de las atribuciones que el Líder del Proyecto haya delegado en ellos;
> 2. pueden tomar ciertas decisiones que el Líder no puede tomar directamente, incluyendo la admisión o expulsión de Desarrollado-

[6] En la Constitución aparecen otras que no es necesario mencionar aquí. Se pueden consultar en Debian.org (s.f.[i]).

> res o nombramiento de personas como Desarrolladores que no mantienen paquetes. *Esto evita la concentración de poder, particularmente sobre la pertenencia al proyecto de un Desarrollador, en manos del Líder del Proyecto*.

Algunos de los equipos que dependen de la delegación por parte del Líder del Proyecto, y que son centrales para el funcionamiento de Debian, son:

- *FTP Masters*, que mantienen el archivo de Debian y los repositorios oficiales de paquetes.
- *Release Team*, que controla el proceso de publicación de una nueva versión de Debian.
- *Press Team*, contacto con la prensa y los medios, así como de las Noticias del Proyecto.
- *New Members Front Desk*, responsable del *New Maintainer Process*.
- *Debian Account Managers* (DAM), responsable de mantener la lista de miembros del Proyecto. Pueden decidir quién es miembro del Proyecto.
- *Debian Keyring Maintainers*, responsable del mantenimiento del conjunto de claves criptográficas que están en los *keyrings*.
- *Policy Editors*, responsable de mantener y actualizar el *Debian Policy Manual* y otros manuales técnicos de referencia fundamentales.
- *Debian System Administrators* (DSA), responsables del manejo y mantenimiento de la infraestructura básica, máquinas y servicios, del proyecto.
- *Debian auditors*, responsables del registro de los recursos de Debian (financieros, de *hardware*, nombres de dominio, marcas, etc.) e informar periódicamente sobre ellos, así como de la relación con las *Trusted Organizations*.
- *DebConf chairs*, responsables de la organización de DebConf, la Conferencia anual de Debian.

En la página *web* «Debian's Organizational Structure» (Debian.org, s.f.[m]) se puede encontrar quién ocupa estas posiciones, así como los correos del DPL donde se establece cada delegación y las tareas encomendadas. Éstas son posiciones que pueden acumular, y de hecho lo hacen, bastante poder de decisión respecto a la marcha y funcionamiento del Proyecto (véase la sección 3.2.4, *Concentración de poder y los límites internos de la doocracia*). Algunas de ellas serán consideradas a lo largo de esta investigación.

3.2.1 Los Desarrolladores Individuales: doocracia.

Veamos ahora cómo trata la Constitución la formalización y explicitación de las decisiones que pueden ser tomadas por los Desarrolladores individuales. En primer lugar hay que señalar una norma general que la Constitución establece como aplicable a cualquier proceso de toma de decisión:

> Nada en esta constitución impone obligaciones a nadie que trabaje para el Proyecto. Una persona que no quiera hacer la tarea que le ha sido delegada o asignada no necesita hacerla. Sin embargo, no se debe actuar activamente en contra de estas reglas y las decisiones tomadas habiéndolas seguido apropiadamente.

Veamos un ejemplo. A finales de 2005, se decidió mediante una Resolución General (Debian.org/vote, 2005) la desclasificación parcial de la lista de correo `debian-private`, a la que sólo tienen acceso los Desarrolladores de Debian. Dado que una parte de los mensajes de esta lista no contienen información que realmente deba ser privada, se entendió que hacerlos públicos favorecería la apertura y transparencia que caracterizan al Proyecto. Sin embargo, no ha llegado nunca a ponerse en marcha porque nadie ha asumido esa tarea, a pesar de la periódica llamada a voluntarios.[7] Tenemos así una decisión tomada por mayoría en una Resolución General que no se implementa porque no hay voluntarios para hacerlo. Y nadie tiene el poder de hacer que se haga. Encontramos esto en la página *wiki* «Declassification of the debian-private archives» (Wiki.debian.org, s.f.[i]), enlazada desde la página web de la votación:

[7] Véase por ejemplo Debian-project (2010a) y la discusión que le sigue.

Debian Private Declassification

Declassification of the debian-private archives

The debian-private mailing list is used for private discussions among developers, which are sparingly needed.

For the sake of transparency, the Debian project voted in 2005 for a declassification procedure which will publish messages posted there after a period of 3 years.

Status of the GR implementation

Debian is more than willing to keep up to its promises and implement what the outcome of this GR requires. Still, the declassification cannot happen per se, but rather needs Debian Developer volunteers to actually *do* that.

Until a suitable team of volunteers with the energy to work on the issue shows up, this GR will remain not implemented.

Posteriormente, en una Resolución General en 2016, se revocó la decisión tomada en 2005, ganando ahora la opción de que la lista permanezca privada (Debian.org/vote, 2016). Este tipo de situaciones ocurre continuamente en Debian, como no podría ser de otra manera en un Proyecto desarrollado por voluntarios. Para que se ponga en marcha cualquier proceso, es necesario que alguien decida libre y voluntariamente hacerlo. Lo cual no implica, como venimos y seguiremos viendo, que el proceso y sus resultados sean totalmente controlables por él. Éste es el punto fundamental para entender la estructura de la toma de decisiones y de la agencia en Debian. Como lo expresa un Desarrollador:

dato: Porque en Debian, básicamente, pasa muchísimo eso. Se sabe que hay muchos aspectos que necesitan atención, que se pueden mejorar, que estaría bien si alguien se ocupara. Pero al final del día, si nadie dice, «yo voy a ir y a mejorar esto», pues nadie lo va a hacer, nadie tiene una batuta que diga «tú, por favor, venga, corre y haz eso», sino que es un poco anárquico en el sentido de que confiamos en la voluntad de los Desarrolladores de gastar tiempo en algo. Y si alguien

3. Desarrolladores individuales
3.1. Atribuciones
Un Desarrollador puede

1. tomar una decisión técnica o no técnica con respecto a su propio trabajo;
2. proponer o patrocinar borradores de Resoluciones Generales;
3. proponerse a sí mismo como candidato a Líder del Proyecto durante las elecciones;
4. votar en las Resoluciones Generales y en las elecciones a Líder.

3.2. Composición y nombramiento

1. Los Desarrolladores son personas voluntarias que colaboran para hacer avanzar los objetivos del proyecto mientras participan en él, y que mantienen paquetes para el Proyecto, o realizan alguna otra tarea que el (los) Delegado(s) del Líder del Proyecto consideren adecuada.
2. El (o los) Delegado(s) del Líder del Proyecto puede(n) elegir no admitir nuevos Desarrolladores, o expulsar a los que ya lo sean. *Si los Desarrolladores consideran que los Delegados están abusando de su autoridad pueden por supuesto anular su decisión mediante la vía de una Resolución General (vea §4.1(3), §4.2).*

3.3. Procedimiento
Los Desarrolladores pueden tomar estas decisiones según crean adecuado.

Cuadro 3.2: Desarrolladores individuales según la Constitución

no tiene voluntad de gastar tiempo en algo, pues no pasa nada, y nadie puede ordenar a nadie que pase. Entonces es un proyecto movido por las voluntades de los Desarrolladores individuales.

Lo que establece la Constitución respecto a los Desarrolladores individuales aparece en el cuadro 3.2, *Desarrolladores individuales según la Constitución*.

Las decisiones tomadas por los Desarrolladores individuales, a diferencia de las tomadas mediante una Resolución General o el Comité Técnico, son tomadas constantemente y no en momentos excepcionales. De ahí que sean las que determinan con más claridad el funcionamiento cotidiano del Proyecto y la concepción predominante sobre la agencia colectiva del Proyecto. Agencia colectiva porque las decisiones que cada Desarrollador toma respecto a su propio trabajo pueden y suelen tener consecuencias para el colectivo en su conjunto. Lo que hace el Proyecto Debian en tanto que tal Proyecto colectivo es, fundamentalmente y en primer lugar, lo que hacen los Desarrolladores por propia iniciativa y voluntariamente.

Como establece la Constitución, no es posible pues imponer «obligaciones a nadie que trabaje para el Proyecto», al tiempo que los Desarrolladores tienen la capacidad de «tomar una decisión técnica o no técnica con respecto a su propio trabajo» y, sobre todo, «los Desarrolladores pueden tomar estas decisiones según crean adecuado». Juntos, estos rasgos configuran una forma de toma de decisiones que los miembros de Debian denominan «doocracia»[8]. De hecho, en una entrevista el por entonces DPL me decía que ésta era la mejor definición de doocracia que conocía.

Tradicionalmente se ha argumentado que la forma de gobierno propia de las comunidades en torno al *software* libre era la meritocracia. Sin embargo, cada vez se usa más el término doocracia, más que como una alternativa a implementar, como una mejor descripción que «meritocracia», o un subtipo de la misma (en tanto se refiere a un mérito específico, el trabajo que se está realizando; así lo indicaba en una entrevista el DPL), de la manera en que de hecho se han venido organizando. En la introducción ya apareció la definición que ofrecía el Líder del Proyecto en 2010: «we operate as a do-ocracy: anyone willing to do things can

[8] De *do*, hacer en inglés, y -cracia, forma de gobierno, poder. El término en inglés es *do-ocracy*. Se suele asociar este término a la organización del festival *Burning Man* (Burningman.org, s.f.), de las *wikis* y de las comunidades de *software* libre. La definición más ampliamente citada es la que aparece en Communitywiki.org (s.f.): «A do-ocracy is an organizational structure in which individuals choose roles and tasks for themselves and execute them. Responsibilities attach to people who do the work, rather than elected or selected officials». No he podido constatar que exista una forma castellana establecida y comúnmente aceptada. He visto usar los términos «doocracia», «hagocracia», «hacercracia», «actiocracia» y «factocracia». Prefiero usar doocracia, que además de ser usado espontáneamente en algunas de las entrevistas realizadas, refleja bien el uso lingüístico muy común entre los Desarrolladores españoles de Debian de usar nuevos vocablos con raíces cuyo origen está en el inglés.

decide what and how they are done, and no one can be forced to do anything» (Debian.org, s.f.[g]).

O como lo expresa una Desarrolladora en un mensaje de correo a la lista de correo `debian-devel-spanish` (Debian-devel-spanish, 2008):

> En ninguna parte de Debian, grupos o no, te van a decir qué es lo que tienes que hacer, eso está claro. En un grupo tampoco. Tienes que ser tú quien veas qué te apetece y puedes aportar. Un grupo es una de las mejores formas de entrar, al menos es mi opinión.

La doocracia se entiende como el mecanismo básico para las decisiones propiamente técnicas. Preguntado por los procesos de decisión para cuestiones técnicas, un Desarrollador de Debian y miembro del Comité Técnico respondía:

> **vorlon:** The general answer is, things that affect the whole Project are the result of an individual package maintainer. Debian has a fairly flat structure, which is useful in doing this kind of thing. We each have our own area of responsibility that we can get work done in, and a package is a very useful unit. Within that, lots of work is done by individual developers, where they do have the final say.
>
> The basic decision process is a really simple one, and sound, it is «the maintainer decides». And it seems to work well.

Por ejemplo, ya vimos a propósito del *bug* sobre `python` como la doocracia es la forma básica de actuación y decisión de los Desarrolladores de Debian. La solución del problema dependió más de la actuación autónoma de los Desarrolladores implicados y de las mediaciones tecnológicas que establecieron que de la decisión final del Comité Técnico o de un proceso más amplio de decisión. Vimos también ahí que existen mecanismos claros que pueden limitar la doocracia cuando es necesario: el recurso al Comité Técnico y la celebración de una Resolución General.

Es decir, la doocracia significa que «el que hace es el que decide» (Hertzog y Mas, 2012, pág. 13). Realizar una contribución o colaborar con el Proyecto no requiere el permiso de nadie, del mismo modo que nadie puede imponer la realización de una determinada tarea. Por otra parte, cada uno es el responsable de sus aportaciones, y tiene derecho a decidir cómo realizarlas. Un elemento a destacar aquí es que esta forma de gobierno facilita enormemente, en su funcionamiento, la aportación de colaboraciones al proyecto y la comunidad. Porque una de las consecuencias de esta forma de organización es que se establece, aunque sea tácitamente, esa obligación de recibir que es parte constituyente del *don* y una de sus condiciones de posibilidad. Veremos esta cuestión detenidamente en el capítulo 5, *El don del* software *libre*. La doocracia, como veremos, es un modo de incluir a los individuos en el colectivo, permitiéndoles hacer cosas por sí mismos. Esta concepción de la doocracia, aquí en estrecha relación con la meritocracia, aparece ya en el «Debian Manifesto»:[9]

> By involving others with a wide range of abilities and backgrounds, Debian is able to be developed in a modular fashion. Its components are of high quality because those with expertise in a certain area are given the opportunity to construct or maintain the individual components of Debian involving that area.

Así, la doocracia se manifiesta también en el proceso mediante el que alguien puede empezar a colaborar y convertirse en miembro del Proyecto (véase el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*). Efectivamente, adquirir el derecho a contribuir, y en definitiva a ser parte de la comunidad de que se trate, no tiene más requerimientos que hacerlo. Una de las consecuencias de adoptar un modelo basado en la doocracia es que se proporciona una vía para la colaboración a cualquiera que desee hacerlo. Como hemos dicho antes, esto aparece siempre en combinación con otras formas de toma de decisiones, y no afecta a todas las partes del Proyecto por igual.

Por ejemplo, una modificación maliciosa o defectuosa de un programa distribuido por Debian realmente pondría en peligro el funcionamiento de muchísi-

[9] En el apéndice C, *The Debian Manifesto*.

mos servidores y ordenadores personales, y por lo tanto sólo pueden ser realizadas (o al menos avaladas) por un Desarrollador de Debian. Cada contribución, cada decisión, aparece entonces ligada a una identidad que determina el acceso a los recursos.[10] Así lo expresa O'Neil:

> Debian must reconcile the central notion of each developer's autonomy, and the respect for difference, with the constraints deriving from the production of a complex system with quality standards of the highest order (O'Neil, 2009, pág. 134).
>
> Contributors to Debian really do have the potential to harm the software. As a result, a central difference between Debian and the other cooperative projects examined in this book is that contributions cannot be anonymous. Online communities are routinely described as having fluid boundaries and shifting members and identities. In contrast, members of FOSS projects who are developing software that is hosted on protected servers connected to the Internet must maintain a distinct and trusted identity, which will enable them to gain access to these protected resources (O'Neil, 2009, pág. 135).

La forma de doocracia que encontramos aquí está entonces limitada a los que son ya Desarrolladores de Debian. Pero en partes del Proyecto menos críticas si está mucho más abierta la posibilidad de colaboración externa. Ésta no se limita a los miembros de Debian. Aunque no esté regulado en la Constitución, en cierto sentido cualquiera puede contribuir al Proyecto a partir de su propia iniciativa, como veremos en el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?* Sobre todo, el proceso para convertirse en Desarrollador de Debian está principalmente determinado por las contribuciones que se han hecho al Proyecto, lo que no es sino una manifestación de esa doocracia. En cualquier caso, tanto los Desarrolladores de Debian como cualquier colaborador está sujeto a las limitaciones que hemos visto, y lo establecido en el «Debian Policy Manual» («Manual de Normas de Debian») (Debian.org/doc, s.f.[d]), que describe las normas técnicas que deben seguir los paquetes de Debian, y la estructura y

[10] Véase la sección 6.4, *La construcción tecnosocial de la confianza*.

el contenido del archivo de Debian, así como algunas cuestiones sobre el diseño del sistema operativo.

En esta sección estamos hablando de Mantenedores individuales, pero lo mismo puede aplicarse a los equipos. Desde luego a los equipos que se dedican a empaquetar *software*, a los que se aplican las mismas consideraciones que a los mantenedores individuales. En este caso, el agente que ejerce la doocracia sería el propio equipo, haciendo abstracción de cómo se organiza internamente. Pero también a otros equipos, como los equipos centrales o *core teams* (FTP Team, DAM, Release Managers, DSA, por ejemplo), de desarrollo o los de *Blends* (versiones especiales de Debian).[11] Como afirmaba el DPL en un *post* (Zacchiroli, 2010d):

> In my recent encounters and contacts with people interested in contributing to Debian, I've found that page to be of invaluable help. In particular, people find it very useful in understanding the macro-structures of Debian and in understanding where they can start to contribute. Approaching a team is most likely «less scary» than contacting a larger forum, and that page offers a good service of team indexing.

En Debian hay por tanto una tendencia a favorecer la creación y funcionamiento de equipos de empaquetado, que pueden solucionar algunos de los problemas que trae consigo la doocracia, y se ven de hecho como solución a diferentes problemas. Ya vimos cómo en la resolución del Comité Técnico sobre el *bug* de `python` analizado recomendaba que hubiese co-mantenedores. En varias entrevistas surgió el tema del trabajo en equipo y la coordinación que se lleva a cabo en el *Debian Perl Group* (Perl-team, s.f.; Wiki.debian.org, s.f.[k]), el equipo que se encarga de empaquetar para Debian módulos de `perl`, un lenguaje interpretado (como `python`). Lo que se destacaba de este grupo por parte de varios Desarrolladores es su estructura horizontal, falta de líder y, sobre todo, la casi total ausencia del sentimiento de que determinados paquetes «pertenezcan» a determinados Desarrolladores. En este sentido, sería un equipo en el que se da mucho menos la tensión entre la *ownership* o propiedad de los paquetes y la doocracia que vere-

[11] Se puede ver una lista de los equipos existentes en Wiki.debian.org (s.f.[j]).

mos más adelante, y donde ésta se ve favorecida precisamente por la ausencia de aquélla. Por ejemplo:

> **tincho:** Hay una política muy importante para el éxito, que es que los paquetes no son de nadie, el paquete lo toca cualquiera, y el que se enoja porque alguien le toca el paquete, está mal, él está equivocado. Está documentado, es política del grupo. Eso es muy importante. Eso es una gran diferencia. He visto en otros grupos donde, [...] vi que la gente, bueno, como este paquete es mío, este es tuyo, y entonces el grupo no tiene mucho sentido. Tiene una lista de correo, tiene un IRC, se ayudan un poco pero no mucho. En el *perl group*, viene uno, sube un paquete, viene otro y ve un error, lo corrige, o si vos estás trabajando mucho en ese paquete, bueno, te manda directamente un mensaje. Es muy abierto al *sponsoring*, por lo que es muy buen lugar para los novatos. De hecho uno de los motivos de PET[12] es justamente todo esto, el PET te muestra en una página *web* todos los paquetes que hay, todos los problemas que tienen, el *changelog*, lo que está haciendo cada persona. [...] Era un excelente lugar para novatos y para integrarse a Debian.

La doocracia necesita en cualquier caso, como veremos, de medios abiertos para su funcionamiento. De hecho, quizás la mejor forma de caracterizar la doocracia sea precisamente esa: la apertura. Que un sistema se rige por esta forma de organización o de gobierno quiere decir también que el sistema está abierto a las aportaciones de quien quiera hacerlo. Si formar parte de la comunidad es contribuir, una comunidad abierta necesita canales abiertos de colaboración. En las secciones 3.3.2, *Apertura y libertad*, y 3.4, *Doocracia y Público Recursivo*, nos detendremos en esta cuestión de la apertura.

3.2.1.1 Doocracia, meritocracia y democracia.

Muchos Desarrolladores hablan de meritocracia en lugar de doocracia. Ésta es una caracterización tradicional de la ética *hacker* (véase la sección 4.2.2, *Me-*

[12] En Pet.debian.net (s.f.). PET fue creado precisamente por este Desarrollador.

diadores e intermediarios. Las figuras del bug *y del* hacker). Así la definen los Desarrolladores de Debian Raphaël Hertzog y Roland Mas:[13]

> Meritocracy is a form of government in which authority is exercised by those with the greatest merit. For Debian, merit is a measure of competence, which is, itself, assessed by observation of past actions by one or more others within the project (Hertzog y Mas, 2012, pág. 12).

Hay que señalar que la meritocracia sigue siendo valorada muy positivamente entre los Desarrolladores de Debian, donde no es infrecuente encontrar valoraciones positivas y reconocimiento de figuras de prestigio dentro de la comunidad que han hecho méritos para ello.

Lo que ambas forman comparten es que sitúan el poder, la autoridad, en el individuo más que en una estructura burocrática, o incluso en una votación, y los hacen depender de la actividad técnica del mismo. Pero mientras una concepción meritocrática sitúa el poder o la autoridad en el interior del individuo *antes* de la acción (dependiendo de su habilidad y de sus acciones pasadas), la concepción de la doocracia que estamos considerando los entiende como *producto* de la propia acción, y los sitúa siempre en relación a sus resultados y a las conexiones que establece con otros dispositivos, más que en su interior. Ésta diferencia se puede observar en lo que me decía una Desarrolladora:

> **ana:** Aquí da igual lo que hayas estudiado, o lo que hayas hecho, aquí [lo que cuenta es] las ganas que tú tengas de hacer cosas, aprender, investigar y hacer lo que tú quieras, que nadie te va a decir, «tú no eres informático y no puedes hacer esto en Debian».

La doocracia permite pensar al actor a partir de las consecuencias de su acción: ésta le constituye más que es explicada por él. Frente a la meritocracia, la doocracia supone que el esfuerzo en mantener las asociaciones de elementos que

[13] En este libro hablan tanto de la meritocracia como de la doocracia.

potencian la agencia ha de ser activamente mantenido, y por lo tanto es menos capitalizable. Las decisiones son tomadas por los que están trabajando activamente en el sostenimiento del entramado sociotécnico.

Una virtud muy interesante del uso analítico del concepto de doocracia es entonces que muestra con mucha claridad el carácter procesual del poder, la capacidad de decisión y la agencia. Éstos surgen efectivamente de lo que se hace, del trabajo entendido como establecimiento de relaciones y asociaciones; en definitiva de la acción social. Doocracia significa que decide el que hace, pero también refiere al poder del hacer, de la acción. Las posiciones de poder no son algo que se tenga previamente, sino algo que surge de los procesos sociales. La doocracia es, en definitiva, un reparto y distribución continuos de la agencia a partir de las acciones y procesos sociales, más que una forma de describir posiciones de poder previamente establecidas.

No se trata por tanto de que alguien consiga algo como recompensa a su trabajo y actividad. La doocracia es una convención, una forma convencional de organizar la acción social que permite a los individuos colaborar, contribuir efectivamente al bien común. Como veíamos al principio del capitulo, es un poder «para hacer tareas, pero no para mandar». No se puede entender entonces sin tener en cuenta que se produce en el marco del *procomún*, de las relaciones de reciprocidad que caracterizan el *don*. Lo veremos en el capítulo 5, *El don del* software *libre*.

Aunque estén siempre en tensión y muchas veces tengan requerimientos y consecuencias opuestos, la doocracia y la democracia no son en absoluto incompatibles. En Debian se entiende que la democracia (como la actuación del Comité Técnico) puede servir como contrapeso y corrección de la doocracia. En una entrevista, un Desarrollador que además era FTP Master y DAM me decía:

> **ganneff:** Debian is a doocracy. So whoever is doing the work does take a decision about it. So if you are maintaining your package you take the decisions on how the package is maintained. [...] If you are maintaining the archive, like FTP Master does, you are taking the decisions on how the archive is maintained. General Resolutions are not used to

> take a decision on how something should be done, because that doesn't work. You can't tell volunteers how they should do things. General Resolutions are used to overrule decisions. For example, if I say that we don't have unstable anymore because FTP Master doesn't want to support it, then the Project can tell me that they want unstable.

Es decir, la Resolución General puede funcionar como mecanismo corrector, pero el proceso básico, por defecto, es la doocracia. Otro ejemplo en este sentido lo podemos encontrar en un mensaje (Debian-project, 2010b) a una lista de correo que escribía el entonces (2010) DPL:

> Debian is not a pure do-ocracy. Rather, it's a «society» (which we call Project) in which do-ocracy can be «corrected» via democracy. Please check the Debian Constitution for the actual mechanisms you can use to trigger Debian democracy. A way in between the two is trying to convince who is doing something to stop doing that, but the default while you try, is that the «thing» they are doing stays there.

Otro Desarrollador expresaba de manera muy clara esta prioridad de la doocracia sobre las votaciones e incluso sobre el consenso. Respondía a una pregunta sobre cómo se ponían de acuerdo en un equipo de empaquetado:

> **bencer:** Es que no hay decisiones en las que todos tengamos que estar de acuerdo. Seguramente las hay, pero creo que hemos borrado ese concepto de nuestra metodología de trabajo, porque si tuviéramos que esperar a estar todos de acuerdo, estaríamos a años luz de donde estamos ahora. ¿Qué se hace? Tomas decisiones cuando estás trabajando. La involucración en los *teams* va muy a ondas, según el tiempo.
>
> El concepto de votación y el de consenso no se toman, y de hecho somos reacios a mandar un correo a la lista: «¿hacemos esto así o así?» No, porque sabemos que va a causar una discusión enorme sin fin. La

> manera en que se hacen las cosas es, he hecho esto así, funciona. Entonces ya vienen los que protestan. Entonces la metodología es decir, bueno, o vienes con algo ya hecho que funciona mejor, o te callas.

En Debian la democracia se manifiesta entonces fundamentalmente a través de las Resoluciones Generales. En este sentido, ambas formas de gobierno pueden efectivamente entrar en conflicto. Pero la doocracia se puede entender también, en sus prácticas, más que como una alternativa a la democracia, como una forma radical de ésta, si entendemos la democracia más en sentido ontológico que procedimental. Si entendemos la democracia como afirmación de la igualdad fundamental de todos, previa a la consolidación de diferentes posiciones de poder. A pesar de que en Debian encontramos una resistencia a utilizar los procesos democráticos de decisión, esto ocurre respecto a los asuntos que llaman técnicos, como vimos, lo que no es incompatible con la afirmación de esa forma especial de democracia que es la doocracia.

Esta relación entre democracia y doocracia se expresaba claramente de la siguiente manera por parte de un Desarrollador en una entrevista:

> **gismo:** In Debian we have this sort of democracy, or as we call it, doocracy, which means that it doesn't matter if you don't know anything. I mean, I am not a good programmer, but every time I make technical comments, if they are good, they are accepted. [...]
>
> In Debian there is liberty, liberty of thinking, of speaking, whatever, and also liberty to do your work. If you don't want to do something, you just say I don't want to do something.

3.2.2 Consenso.

¿Qué ocurre entonces cuando las decisiones tomadas por dos o más desarrolladores chocan o son incompatibles? Se puede acudir a una Resolución General, y ya vimos cómo el Comité Técnico funciona como un órgano de apelación en estos casos. También vimos que según la resolución del Comité Técnico sobre el *bug* de `python` analizado la mediación entre diferentes personas había fallado, y

se pedía que en el futuro se produjese ésta, por parte de miembros ampliamente respetados, en los conflictos. Éste es de hecho uno de los papeles del DPL. Pero siempre es preferible llegar a un consenso antes de que se produzca un conflicto, como se plantea también en varios lugares de la Constitución. De hecho, en las entrevistas realizadas el consenso aparece en numerosas ocasiones como la forma preferida de coordinación, y todos los entrevistados muestran una gran confianza en el mismo para guiar y resolver las discusiones públicas. Así se expresa una Desarrolladora:

> **marga:** Pienso que es importante cambiar la forma de discutir las cosas en Debian. O sea, poder tener una discusión civilizada, tratar de llegar a un consenso y, bueno, si no se da el consenso buscar cuáles son las posiciones básicas y hacer una votación. Pero ha pasado que la gente agarra y empieza por la votación, sin primero charlarlo y ver si se llega a un consenso o no, y si no se llega a un consenso ver cuáles son realmente las posiciones que hay. Como que ya arrancan por la votación, sin primero pasar por ese periodo de discusión sana.
>
> [...] está la idea de las DEPs justamente para tomar este tipo de decisiones, que en realidad no necesariamente son sociales. Para digamos, no tener que estar pasando por un proceso electoral, tratar de llegar a un acuerdo consensuado sin el proceso electoral. La verdad es que mucho no prendió, están como hace tres años las DEPs y no llegamos a la número diez.

Esta Desarrolladora plantea esta cuestión a propósito de una decisión sobre los procesos para llegar a ser parte de Debian tomada por el *Debian Account Manager* que analizaremos en el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?* Señalo esto porque esa decisión se presentó en un primer momento como una decisión que afectaba al Proyecto en su conjunto, pero tomada por un miembro en virtud del puesto que ocupaba, de acuerdo con la doocracia que caracteriza al Proyecto. De hecho, esto es un motivo de tensión recurrente en el Proyecto. Son varias las decisiones con las que muchos miembros posiblemente estén de acuerdo, pero se produce un rechazo por ser tomadas antes de alcanzar un acuerdo amplio. Además de la citada, se pueden mencionar el

anuncio, en julio de 2009, por parte del *Release Team* de modificar el proceso de publicación de una nueva versión del sistema operativo, adoptando un esquema previsible en el tiempo.[14] Otra fue el anuncio del proyecto Dunc-Tank (lo sustantivo de esta cuestión se expone en la sección 5.1.3, *Trabajo voluntario y trabajo remunerado: tensiones y controversias*). En este caso definitivamente, además, no había consenso:

> **marga:** La otra terrible fue la del Dunc-Tank. Fue algo similar. Se planteó una idea, pero en vez de plantearlo así, busquemos un consenso al respecto, se planteó esa idea, se va a hacer, y perdimos muchas fuerza en Debian por culpa de eso. Mucha gente que renunció a Debian por culpa de eso, y mucha gente que no renunció pero disminuyó el nivel de tiempo, de esfuerzo y de compromiso en Debian por culpa de eso.
>
> **Fernando:** ¿Por cómo se planteó o por el contenido?
>
> **marga:** Creo que un poco y un poco. El hecho de plantearlo, lo vamos a hacer en vez de buscar la forma y que todos estuvieran de acuerdo. Y también por el contenido, [...] y no se supo decir que si el consenso es que no, entonces no lo hacemos. Se hizo igual cuando el consenso era que no. [...] Al principio no era tan claramente que no, pero era una cosa en la que había varios que se oponían mucho. Cuando hay varios que se oponen tanto es como que es algo que hay que ..., o sea cuando se te van cinco de los Desarrolladores principales, bueno, pará un poco, pensá un poco en lo que estás haciendo.

Sobre el mismo ejemplo, otro Desarrollador pone claramente de manifiesto algunos de los problemas a los que puede llevar esta tensión entre mecanismos de decisión. Además de los problemas que supone no alcanzar un consenso, vemos claramente aquí el poder que tienen los Desarrolladores individuales:

[14] No podemos desarrollar aquí esta interesante controversia. El anuncio se puede encontrar en Debian-announce (2009), y buena parte de la discusión en el hilo que sigue a Debian-project (2009b).

> **guillem:** Hay veces que se implementa [una decisión] y se notifica después del hecho. El problema con eso es que si la gente lo ve bien o es un diseño limpio y no ve ningún problema, perfecto. Pero cuando hay problemas entonces es mucho peor porque la gente se indigna de que no se ha notificado previamente.
>
> Un ejemplo a nivel global de proyecto fue Dunc-Tank, el tema de pagar a los *Release Managers*, eso fue una implementación que se propuso como una cosa que se iba a hacer [...] Yo creo que fue un fracaso por parte de la iniciativa porque creó muy mal rollo en el proyecto. El *Release Team* necesita del cuerpo de Desarrolladores para hacer una *release*, y hubo gente que activamente estuvo impidiendo el ..., o sea no por el hecho de destruir el Proyecto, pero por ejemplo había gente que podría haber evitado hacer cosas que impedían la *release* pero las hicieron. Había gente que se dedicaba a recompilar todo el archivo para buscar paquetes que fallaban al compilar o detalles, cosas, no sé, casos específicos, pues buscar problemas en el archivo en general y mandar cientos de *bug reports*, pero eso claro afecta a cuándo se va a poder hacer la *release*.

Lo que se busca pues no es sólo una manera de organizar el consenso, también de evitar o suavizar algunos problemas de la doocracia. Aunque el consenso es por su propia naturaleza difícil de formalizar, como señala el fragmento de entrevista con el que abrimos esta sección, el Proyecto Debian ha implementado un mecanismo para organizar el consenso sin tener que recurrir a votar cada asunto, y sin dejarlo exclusivamente a la imprecisión de los procesos de discusión en las listas de correo o los canales de IRC, las *Debian Enhancement Proposals* o DEPs (Dep-team, s.f.[a]):

> This is a proposal to organize discussions about Debian enhancements, reflect their current status and, in particular, to archive their outcomes, via a new lightweight process based on Debian Enhancement Proposals (DEPs). This idea is loosely based on existing similar systems such as RFCs and Python PEPs. It is also completely opt-in, and does not involve any new committees, powers, or authorities.

Su motivación es aclarar cuando se ha alcanzado efectivamente un consenso:

> Currently, when having discussions about improvements to Debian, it is not always clear when consensus has been reached, and people willing to implement it may start too early, leading to wasted efforts, or delay it indefinitely, because there's not clear indication it is time to begin.

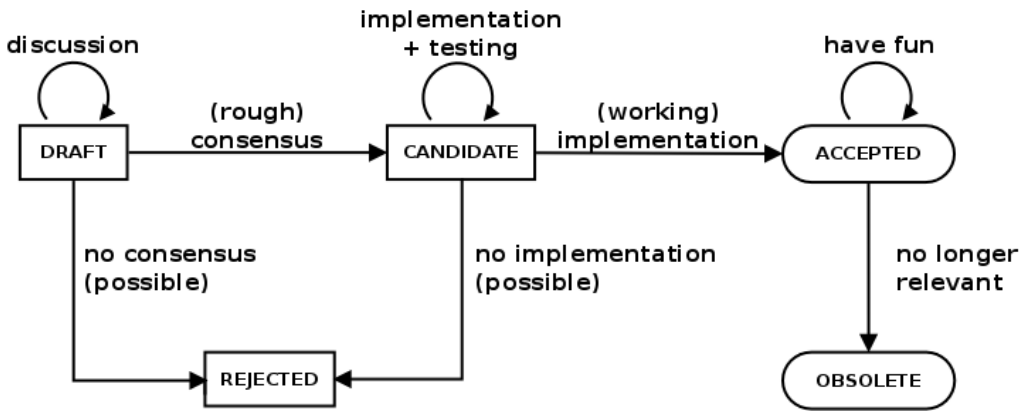
Puede usarse este dispositivo para promover cualquier cambio en Debian, técnico o no. Pero no se trata de crear todavía otro foro de discusión, sino que la discusión debe mantenerse en los ya existentes, fundamentalmente las listas de correo y el IRC. No se trata de establecer una nueva autoridad, ni otro canal de comunicación, sino de organizar el proceso del consenso:

> In the same way, DEPs do not give any extra powers or authority to anyone: they rely on reaching consensus in the traditional Debian way, by engaging in discussions on mailing lists, IRC, or real life meetings as appropriate, and not by consulting an external body for a decision. To be acceptable, this consensus includes agreement from affected parties, including those who would have to implement it or accept an implementation.

En el cuadro 3.3, Debian Enhancement Proposals Workflow, tomado de esa página *web*, se puede ver el flujo de trabajo para proponer, discutir y en su caso aprobar o rechazar las DEPs.

Sin embargo, la realidad es que no se usan demasiado. La primera (que consiste precisamente en la propuesta de las DEPs) data de 2009, y actualmente (verano de 2018) hay 15 en diferentes estados de desarrollo, como se puede ver en su página *web*.

Por otra parte, los procesos para llegar a consensos no están libres de problemas. No es posible evitar la tensión entre el deseo de horizontalidad, la falta de



Cuadro 3.3: *Debian Enhancement Proposals Workflow*

jerarquías y el consenso por una parte, y la capacidad de decidir autónomamente sobre los paquetes que uno mantiene, tal como se deriva de la doocracia, por la otra. El conflicto entre doocracia y consenso es inevitable. El Desarrollador (y en esa época miembro del Comité Técnico) de Debian Russ Allbery explicaba así en su *blog* los problemas de un proceso de decisión basado en el consenso (sobre todo en grupos muy numerosos), y las ventajas de un sistema como el de Debian (Allbery, 2013):

> [...] And on the surface, it may look like Debian also uses a consensus-based decision-making process. But while Debian decision-making has some problems, it's much healthier for a few reasons that are useful to examine:
>
> 1. Individual packages in Debian are not managed by consensus, but rather fall under the authority of the package maintainers [...]. Nearly all decisions in Debian are made directly by the person doing the work, and only in exceptional circumstances can some larger process be invoked.
>
>   [...] There is a strong bias in favor of letting the person working on the package do their job however they want to do it. This is more social than structural, but it's a core value (and we should be very careful about changing it).

As a result, Debian operates less by mass consensus and more as a federation of semi-autonomous fiefdoms. [...]

2. On appeal, and for broader questions that necessarily cut across multiple packages, there are several people in Debian who are perceived as having clear authority to make timely and relatively final decisions, who are almost never overruled, and who can therefore effectively end arguments. These include core teams like ftp-master or the release team, formal appeal bodies like the Technical Committee, and the Debian Project Leader. [...]
3. There is a well-established ultimate appeal (a General Resolution), the ultimate appeal process is very specific and concrete, there is little or no ambiguity or human interpretation involved in analyzing the results, it is time-limited, and it's almost universally respected. Everyone abides by the result of a GR if matters get to that point, and while it's theoretically possible to propose another GR immediately to overturn the first one, this doesn't happen in practice. There's also substantial pushback against invoking the GR process for anything that isn't really important.

3.2.3 Problemas de la doocracia: *ownership*, *hijacking* y *salvaging* de los paquetes.

La doocracia proporciona entonces un mecanismo operativo, funcional y rápido para tomar la mayoría de las decisiones. El problema del consenso como mecanismo de decisión es que no existe un procedimiento para terminar la discusión si alguna de las partes persiste en la discusión. Pero no todos los miembros de Debian están de acuerdo. Así le contestaba al *post* anterior el también Desarrollador de Debian Clint Adams (2013):

After recovering from the shock of seeing "consensus" spelled correctly for a change, I thought I should respond to some claims of healthiness.

1. Package fiefdoms waste people's time quite frequently: MIA processing, duplicated effort in DELAYED NMUs, conspiracy and infighting, trying to get an absentee maintainer to respond, the

> ridiculous things that happen when a maintainer is still around but not doing his job for 3 consecutive years, and so on. Having all packages maintained by "Debian Developers" is a great idea.
>
> 2. The current hierarchy might work better than some lesser evils, but that doesn't mean it's good at all. We should strive toward ZERO hierarchy, not some magical fiction where people imagine themselves wiser than everyone else because their drinking buddies said so.
>
> The anti-consensus features in Debian would better be replaced by a real solution to the problem pervading nearly all free software projects: ego.

En el *post* citado, «Debian Developers» es un enlace a un post del Desarrollador Dimitri John Ledkov, en el que propone precisamente que no haya «propiedad» individual de los paquetes, sino que estos sean colectivamente mantenidos bajo ese nombre, con los mismos derechos para cualquier Desarrollador de Debian (Ledkov, 2013):

> **shared maintainership**
>
> I'd like to put all my packages under «Debian Developers» maintainership. There is no need for NMU, QA, TEAM upload. If you have a patch, I don't want a bug in BTS nor commits, I'd like you to simply upload it into the archive.

Por lo tanto, vemos que uno de los aspectos más cuestionados de la doocracia son los problemas relativos a la propiedad (*ownership*) de los paquetes. Se produce una tensión entre el derecho adquirido *de facto* a mantener un paquete como cada Desarrollador crea adecuado, y el reconocimiento del derecho de cualquiera a contribuir a ese desarrollo. Este privilegio se obtiene cuando se sube un paquete al archivo por primera vez. No es un derecho formal y hay formas de corregirlo, pero es de hecho uno de los presupuestos de la doocracia. La tensión se produce también, como veremos más adelante (capítulo 5, *El don del*

software *libre*), entre lo que se dona y lo que se guarda, entre la doocracia y la obligación de recibir. Ya hemos visto una situación, la discusión sobre `python`, en la que se produjo una disputa sobre la «propiedad» de determinados paquetes. En una respuesta al informe del *bug* analizado, un Desarrollador expresaba esta tensión (Bugs.debian.org, s.f.[f]):

> We can pass the message that no matter how you maintain your packages, no matter how badly you behave with your peers, no matter what you do, you'll be grant forgiveness, and we'll pretend all of these never happened and you'll be left in an advantage position over maintaining your packages (even for the fear of possible repercussions in other activities for Debian).
>
> Or we can pass the message that we are a community (and not just a bunch of geeky people) and we must act like one, so if you're deliberately screw with your peers because you don't care for them, if you feel so superior to them that you're not going even to speak to them, if your interest in packages maintenance is so low and several times you've uploaded «half-beaked» packages failing even the basic pre-upload tests (like installing them) then we'll bring the packages back to the community, where they belong to, and given in hands proven to caring for them.

Encontramos otra situación similar en la discusión sobre la «propiedad» de la «Debian Developer's Reference» (Debian.org/doc, s.f.[e]), un documento que explica los procedimientos recomendados y los recursos disponibles para los Desarrolladores de Debian, y por lo tanto un documento fundamental para el Proyecto. La discusión se encuentra en el *log* del informe de fallo al paquete `tech-ctte` #436093 (Bugs.debian.org, s.f.[g]), cuyo asunto es «Please decide on the "ownership" of the developers reference».

Lo interesante en esta discusión es la concepción que podemos descubrir de la *ownership* como acceso con privilegios a un repositorio y la capacidad de hacer *commits* (véase la sección 4.8, *Repositorios de* software *libre*), es decir, el permiso de cambiar los archivos del paquete. El conflicto se produce en torno al derecho

a introducir cambios directamente en el documento, sin pasar por la aprobación previa de quien se ha establecido como mantenedor del mismo. Sin entrar en los detalles de la discusión, sólo quiero señalar aquí que, debido a la doocracia, se producen en Debian efectivamente tensiones entre el derecho adquirido a ocuparse de determinados paquetes y tareas, aunque de hecho todos los Desarrolladores de Debian pueden técnicamente subir paquetes al archivo de Debian, y el carácter colaborativo del trabajo en el Proyecto Debian. Un Desarrollador menciona así algunas ventajas e inconvenientes:

> **dato:** En Debian, desafortunadamente, ha existido desde siempre, quizás ahora menos, pero todavía, el concepto de *attachment* de cada Desarrollador a sus paquetes: esto es lo mío, mi reino, y nadie más lo toca [...] Tiene ventajas e inconvenientes. Yo creo que es bueno que se vea bien claro quién es el encargado de un paquete, porque si no se pierde un poco el sentido de la responsabilidad. Pero me gustaría que existiera más la predisposición de que si alguien viene y te quiere ayudar, aunque sea puntualmente, pues pueda ayudarte y que no suponga un problema.

En Michlmayr y Hill (2003), dos Desarrolladores de Debian exponen algunos problemas y ventajas de que el desarrollo y el mantenimiento de paquetes sea realizado por un Desarrollador individual frente al trabajo en equipo, desde el punto de vista de la calidad del *software* distribuido. Si bien es menos fiable su continuidad, dado el carácter voluntario del trabajo, esta forma de organización tiene la ventaja de que no son necesarias estructuras complejas de decisión y se limita la complejidad de los mecanismos de colaboración y comunicación. Además, el sentido de *ownership* respecto a un paquete contribuye a aumentar la motivación e implicación del Desarrollador individual.

En la DebConf de 2012 hubo un BoF[15] llamado «Hijacking packages for fun and profit».[16] El organizador mandó un resumen del mismo a la lista de correo

[15] *Birds of Feather*, una reunión informal. Véase la sección 6.2.1, *DebConfs*.

[16] Hay más información en Penta.debconf.org (s.f.[a]). El vídeo puede verse en Meetings-archive.debian.net (s.f.[b]).

`debian-devel` (Debian-devel, 2012a), donde aparece un planteamiento muy claro de uno de los problemas que se plantea sobre la propiedad de los paquetes:

```
To: debian-devel@lists.debian.org
Subject: «Hijacking packages for fun and profit» BoF at DebConf
From: Steve McIntyre <steve@einval.com>
Date: Fri, 20 Jul 2012 19:24:15 +0100

Hi folks,

Here's a summary of what we discussed in the BoF [1] last week (12th July).
Thanks to the awesome efforts of the DebConf video team, the video of the
session is already online [2] in case you missed it. I've also attached
the Gobby notes that were taken during the session. I'd like to express
my heartfelt thanks to the people who took part - we had a very useful
and productive discussion on a potentially very controversial topic. It
would be good to continue the discussion with a wider audience.

Package maintainership and ownership
====================================
The concept of package maintainership is central to Debian. It's one of
Debian's biggest strengths, in that we can delegate technical decisions
about packages to expert individuals or small groups of maintainers. They
can then do their work and make things happen without having to seek approval
of the whole project for every change they make.

The flip side of this is that package maintainership can also be a problem:
maintainers can be too territorial about «their» area and block development
when they're inactive or disagree with proposed changes. It can often get
in the way of the «do-ocracy»; if a maintainer does not have the time or
inclination to work on their package, they can still stop other people
from doing it.
[…]
```

Aparece claramente formulada la idea de que ese sentimiento de propiedad puede dificultar la doocracia, impidiendo que otros trabajen en el paquete. Pero es también producto de la misma doocracia, como se muestra en el correo: «we can delegate technical decisions about packages to expert individuals or small groups of maintainers. They can then do their work and make things happen without having to seek approval of the whole project for every change they make.» Una forma de limitar o neutralizar los problemas de esa propiedad son los *Non Maintainer*

«A non-maintainer upload (NMU) is an upload of a package to the debian archive by a developer who is not the maintainer of the package. This should usually not be the case, but in special cases (such as for RC bugs, when the maintainer does not respond to the bug report) it is allowed» (Wiki.debian.org, s.f.[l]).

No todos los Desarrolladores aceptan igualmente bien que se realice un *NMU* de los paquetes que mantienen, puesto que, de alguna manera, esta posibilidad entra en tensión con la «propiedad» de un determinado paquete y la doocracia que otorga a cada Desarrollador el derecho a decidir sobre ellos. Aquellos que declaran aceptar que los NMU a sus paquetes se realicen fácil y rápidamente aparecen en Wiki.debian.org (s.f.[m]). Tras una propuesta de mejora (DEP; véase en Dep-team (s.f.[b]), las recomendaciones para realizar un NMU se encuentran en la sección 5.11 de la «Debian Developer's Reference» (Debian.org/doc, s.f.[e]).

Cuadro 3.4: *NMU*

Uploads, o *NMU* (cuadro 3.4, *NMU*). Esta práctica establecida consiste en subir una versión nueva de un paquete al archivo de Debian por parte de alguien que no es su mantenedor habitual, limitando así efectivamente el derecho exclusivo de un desarrollador a mantener el paquete como decida. Por esta razón no se solían usar mucho y muchas veces estaban mal vistos. Eso es algo que ha ido cambiando, como explicaba un Desarrollador (Zacchiroli, 2010h):

> Additionally, RCBW is meant to dispel the old Debian folklore that «NMUs are bad», quite the contrary: NMUs are good and helpful. In the old days of tight package control, we've grown accustomed to strong package ownership; according to that culture doing a NMU can be seen as a personal insult towards the current package maintainer. Nowadays things have changed: Debian is bigger, we routinely work in teams, and we have hard time spotting de facto MIA/inactive maintainers. Also we have delayed NMUs and appropriate guidelines that avoid the risk of impromptu uploads, when followed thoroughly.

Los NMUs se deben usar para cambios urgentes o como solución temporal, como corregir *bugs* críticos, pero no deben modificar quién es el mantenedor del

paquete. Se espera que el mantenedor siga siendo el mismo y que integre los cambios hechos al subir al archivo su propia versión. Si los cambios van más allá y modifican elementos no urgentes del paquete, o cambian el mantenedor del mismo, se habla entonces de secuestro o apropiación (*hijacking*) del paquete. Son normales las controversias sobre si un NMU está justificado o los cambios han ido demasiado lejos.[17] Si un Desarrollador cree que un paquete debe cambiar de mantenedor porque éste no lo mantiene de manera apropiada o está desaparecido, se puede pedir que se declare huérfano (un paquete huérfano es el que no tiene mantenedor) y sea así adoptado por otro mantenedor, o que sea eliminado del archivo dando así la oportunidad de que otro Desarrollador lo suba. A raíz del BoF y la discusión en la lista mencionados, hubo dos grandes hilos en `debian-devel`[18] sobre la formalización de un proceso que permita dejar huérfano un paquete mal mantenido. Aunque no se formalizó como tal la propuesta, la longitud de los hilos de discusión muestra el interés de la cuestión. En la sección 7.4 de la «Debian Developer's Reference» (Debian.org/doc, s.f.[e]) se puede encontrar el modo de solicitar que un paquete se deje huérfano si está mal mantenido.

En el último mensaje citado se propone la distinción, surgida en el BoF, entre secuestrar o apropiarse de un paquete (*hijacking*) y salvarlo o rescatarlo (*salvaging*):

> What is hijacking?
> ==========
> There are cases where we need to balance maintainer control against wider requirements. Two different issues here.
>
> - taking over a package where the existing maintainer agrees or is missing/MIA
> - taking over a package where the existing maintainer objects
>
> To help distinguish them, let's call the first of these «salvaging».
>
> Salvaging
> If an existing maintainer seems «clearly» not to be maintaining a package, then it should be simple to assume maintainership. Mail that maintainer asking if they would be happy with this, and give a reasonable length of time (suggestion: 1 month) to respond. If (ideally) they respond positively or (less ideally) there is no response, then it should be considered sensible

[17] Como ejemplo puede verse la discusión que sigue a Debian-devel (2010).
[18] Véanse Debian-devel (2012b) y Debian-devel (2012c), y los mensajes que les siguen.

> to take over the package. If the existing maintainer objects, then continuing on becomes a hijack.
>
> The explicit concept of **salvaging** seemed to be new in this discussion, and was generally agreed to be a good way of thinking about the problem.
>
> Hijacking
> If there is continued disagreement over who should be the maintainer here, or (more generally) how maintenance should be done, then rather than simply argue endlessly and cause bad blood a good option should be to take it to the Tech Committee for a ruling; they are the correct body to arbitrate here. Ian was very much in favour of this, even if he was worried the rest of the TC might be less happy… :-) There was also talk of a GR to explicitly ask the TC to take more aggressive control in this area.
> […]

El *hijacking* es una práctica en general mal vista y desaconsejada. Si se hace de algún modo necesaria, se recomienda en su lugar acudir al Comité Técnico. En cambio, *salvaging* es algo que se considera positivo.[19] En esa diferencia de términos se resume una de las tensiones producidas por la doocracia, entre el derecho a mantener el paquete como el Mantenedor decida, y la necesidad de abrir los paquetes a la colaboración. Es la doocracia la que da ese derecho al Mantenedor ya establecido, pero también la que se ve perjudicada por el bloqueo de un Mantenedor que no hace lo que se espera de él. Vimos en el capítulo anterior cómo el Comité Técnico era quién podía cambiar oficialmente el mantenedor de un paquete. Pero es una opción que de hecho no se usa. En diciembre de 2016 encontramos una discusión sobre esta cuestión. No vamos a entrar en su análisis,[20] pero en ella se encuentran diferentes concepciones sobre las ventajas e inconvenientes de esta manera de entender la «propiedad» de los paquetes, y diferentes propuestas que van desde su eliminación total a una distinción entre *strong* y *weak ownership* y la intención de pasar de la primera a la segunda.

Por último, podemos mencionar muy brevemente otro de los problemas de la doocracia: a la hora de publicar una nueva versión del sistema operativo, no hay, como me decía un Desarrollador de Debian, una figura central con capacidad

[19] Se puede seguir parte del desarrollo posterior de esta cuestión a partir de los mensajes a la lista `debian-devel` (Debian-devel, 2016a, 2018).

[20] La discusión sigue al mensaje Debian-project (2016a). Algunos mensajes interesantes del hilo son Debian-project (2016d,b,c).

de decidir qué cambios han de realizarse en determinados paquetes.[21] Sí existe un equipo de coordinación, el de los *Release Managers*, con capacidad de fijar objetivos, plazos y procedimientos en este proceso, pero que dependen también de la actividad de cada Mantenedor. Esto, entre otras cosas, hace que las nuevas versiones de Debian se produzcan en plazos largos y algo imprevisibles, algo que por otra parte es motivo de orgullo para muchos de los Desarrolladores de Debian en tanto que es una señal de que se prima la calidad por encima de todo. En Debian se repite continuamente que se publica una nueva versión del sistema operativo cuando está lista, no antes.

3.2.4 Concentración de poder y los límites internos de la doocracia.

Tenemos entonces dos grandes tipos de procesos de toma de decisiones en la estructura organizativa en el Proyecto Debian. Por una parte, aquéllos que están claramente formalizados en la Constitución, con reglas claras que rigen su puesta en práctica: las votaciones democráticas de las Resoluciones Generales, las decisiones del Comité Técnico, o las decisiones que el DPL, sus Delegados o el Secretario pueden tomar en virtud de su cargo. Por otra parte, la capacidad de cada Desarrollador para actuar respecto a su parcela de trabajo, especialmente el mantenimiento de paquetes, como estime conveniente. Ésta es con mucho la forma de tomar decisiones más frecuente y la que configura las prácticas cotidianas de los miembros de Debian. Esta capacidad está también reconocida y delimitada en la Constitución, pero como hemos visto su funcionamiento y consecuencias son más complejos de lo que allí se establece.

Este modelo en el que los Desarrolladores tienen su parcela de poder adquirido *de facto* (sin olvidar las limitaciones y contrapesos que hemos visto) respecto

[21] En 2015, el entonces DPL Lucas Nussbaum señalaba algo parecido respecto a la gran controversia sobre la elección del sistema de inicio por defecto de Debian, que no hemos tratado aquí (véase la sección 1.3, *Cuestiones de metodología*): «The init system debate has put a lot of pressure and stress on Debian, as the decision was not only technical, but also political: it was not only about choosing the best init system at a specific point of time, but also a question about how to move forward for the coming years. The fact that it was also a situation where no package maintainer was clearly responsible for making the decision was also quite unique in the history of Debian. In retrospect, I do not really think that this could have been handled much better: it is just the kind of decision that is hard to make for organisations like Debian that value making correct and well-informed technical decisions» (Varghese, 2015).

a lo que hacen funciona bien y es generalmente aceptado respecto al mantenimiento de paquetes de *software*. De hecho, para muchos es un requisito fundamental para el trabajo colaborativo descentralizado. Pero en ocasiones los miembros de Debian manifiestan también su preocupación ante el hecho de que puede causar concentración de poder cuando se aplica en otras situaciones. Así es como aparece el poder de los *Debian System Administrators*, los *FTP Masters* o los *Debian Account Managers*.[22] Y el principal mecanismo de compensación y limitación es la confianza en que este poder se usará bien y con responsabilidad. Es un tipo de estructura que permite a determinados individuos determinados privilegios, que dependen fundamentalmente del acceso privilegiado a determinados recursos del proyecto y su infraestructura técnica.

El control del acceso a determinados servicios o máquinas se convierte entonces en un límite interno de la doocracia. La limita en tanto que se puede impedir que alguien, Desarrollador o no, pueda hacer lo que crea más conveniente. Pero es un producto de la misma, en tanto que quien tiene la potestad de decidir sobre esas cuestiones es quien efectivamente proporciona y mantiene esos servicios. Nos encontramos aquí con una concepción del poder y la autoridad como *control del acceso*. Tiene poder quien es capaz de otorgar o denegar el acceso a servicios, máquinas o recursos.

Respondiendo a una pregunta sobre la distribución de poder en Debian, sobre si ésta consiste en una estructura horizontal, sin jerarquías, o sí existían grupos con más poder, una Desarrolladora respondía así:

> **marga:** Yo creo que ahora hay mucho menos que antes. Antes había tres o cuatro que controlaban digamos los puntos críticos de casi todo. Ahora está mucho más repartido, si bien ganneff en particular tiene mucho poder. [...] o sea poder sería como entre comillas, o sea él tiene acceso, control, más que poder sería control, sobre un montón de cosas, pero labura una barbaridad, o sea le dedica una cantidad de tiempo a Debian. [...] Tiene sentido que tenga mucho espacio. Yo veo positivo que en los últimos tres años se ha repartido mucho más, y no

[22] Sobre esta cuestión y lo que sigue véase Coleman (2013, cap. 4, esp. pp. 135-140).

está todo tan puntual en tres o cuatro personas. Para mí eso es algo que ha mejorado.

Por su parte, otro Desarrollador se expresaba así respondiendo a la cuestión de si Debian era una meritocracia o una doocracia:

clint: Well, people used to say Debian is a meritocracy, and it is not. And now they say it is a doocracy, and it is not. It is not a meritocracy because you get power in other ways than showing merit, and it is not a doocracy because only certain people have the power to do. And unless those people give you the power to do you can't do, so if you can't do you can't have the power.

So, I think the way it should be, is that we should be as equals as possible, we should all have the same amount of power. [...] Many people are hostile to the idea of a flat power structure, they want to move towards more levels, more granularity [...] Many people, if you give them the access to improve something, [...] they will do more than if no one has the access.

Un desarrollador de Debian respondía así a la cuestión de si la existencia de personas o grupos con más privilegios de acceso técnico creaba alguna dinámica de concentración de poder:

guillem: Yo creo que sí, es una cosa que siempre ha ocurrido. El problema es que en Debian tienes un grupo muy grande de gente que [...] mantienen un paquete o dos [...] muy focalizados. Después hay un grupo bastante pequeño que tiene intereses globales sobre el proyecto, [...] al final es como el grupito conocido de gente que acaba haciendo un poco de todo. Entonces llega un punto que los poderes se acaban acumulando en el grupo central. [...] Yo creo que es normal que la gente acumule más roles de los que puede manejar, pero sí, es un problema, yo creo que ha pasado siempre desde que estoy involucrado en Debian

> que los temas de infraestructura han estado manejados por un grupito pequeño que en este momento concreto tiene responsabilidades en varios grupos centrales.

La Desarrolladora que acabamos de citar identificaba así las posiciones dentro de Debian en las que se concentra más poder y capacidad de decisión:

> **marga:** están los DAM (*Debian Account Manager*), que son los que dejan entrar a gente. [...] Pero el *keyring* no lo manejan ellos, lo manejan [otros Desarrolladores]. Eso es algo importante, porque cuando yo entré en Debian, el *keyring* y las cuentas, lo manejaba todo James Troup, y si le caías mal no entrabas. Y era, sí, era «el» paso. Ahora hay dos DAM, dos *keyring maintainers*, que son personas distintas.
>
> Por otro lado está el *FTP Team*, que son los que dejan entrar a los paquetes. Ahí está ganneff también, pero también hay unos cuantos más, y desde hace tiempo ganneff viene intentando reclutar más gente para el *FTP Team*. [...] han incorporado bastantes personas, han ido rotando gente [...] eso para mí está muy bueno, que hayan podido reclutar más gente y no se hayan quedado en su grupito de tres personas.
>
> Y después el otro grupo son los que mantienen las computadoras de Debian [*Debian System Administrators*]. También ese grupo se ha diversificado un poco, en su momento eran tres, y como que eran sólo esos tres, la élite máxima de los *maintainers*, y ahora se amplió un poco el espectro.

Este fragmento es muy revelador. Las posiciones en las que se podría decir que se concentra el poder en Debian son, por un lado, quienes mantienen los ordenadores de Debian, su infraestructura técnica. Por el otro, quienes en última instancia controlan el ingreso de paquetes y de nuevos miembros. Es decir, los que controlan quién forma parte del colectivo sociotécnico, que como veremos en la sección 6.1, *De la comunidad al colectivo sociotécnico*, incluye dispositivos tecnológicos tanto como personas. Estos poderes no están como tales recogidos explícitamente en la Constitución. Dependen de la Delegación que hace el Líder del

Proyecto, pero sus consecuencias surgen de la exclusividad del acceso a los recursos del Proyecto. De hecho, durante mucho tiempo, y antes de empezar el trabajo de campo, estas posiciones no eran objeto de delegación por parte del DPL, y por lo tanto no estaban reguladas por la constitución, siendo anteriores a su aprobación.[23]

En una entrevista con un Desarrollador de Debian aparece con mucha claridad esta concepción del poder como acceso. La enumeración de las posiciones con más poder en Debian es similar a las que ya hemos visto:

> **micah:** Well, the DSA, the system administrators, they control all the Debian machines, so the domain name, the email, the websites, all of these, they have a lot of power. The FTP Masters have a lot of power, because they can remove packages and allow packages in, or they can deny you the ability to upload a package. The Release Managers have quite a bit of power also, because they are directing the release and they make choices, and have some access. And I would say also the Debian Account Manager position.

E indica que es efectivamente el acceso a los diferentes recursos y servicios del Proyecto donde radica esta concentración de poder:

> **micah:** Yeah, the access to the machines, like the security or the system administration people have root access to the machines, and they control the registration of the domain name. So if they wanted to, they could take control of the whole project, and cut everyone off. I cannot do that, because I don't have that access. So they have some more power to control or stop people from doing things or say, we are taking

[23] Un momento clave en esta historia, tal y como me fue referido en algunas entrevistas, fue la delegación, por parte del DPL Sam Hocevar, de ganneff como FTP Master y DAM, junto al anterior Anthony Towns (que también había sido DPL) y en contra de los deseos de este último, en 2008. Esta situación provocó la renuncia de Towns a sus responsabilidades en Debian. Este cambio produjo una mejora de la situación de bloqueo y concentración de poder percibida por muchos Desarrolladores. Al respecto puede verse Wülfing (2008).

> everything and we are doing something else with it. While the other developers don't have this ability, because they don't have the access [...] So it's, you know, root access or more privileges on the systems. Some of the teams have different privileges that allow them to do things that other developers cannot. So that's pretty much what it is, they can direct these resources in whichever way they decide. Everybody expects them to do the right thing, and they do. But if they decide not to, they can go that path. But if I decide not to do the right thing, I'm very limited in what damage I can do. I can upload a package that is maybe a package that screws something up, but that's it, it's very limited.

Mientras las cosas funcionan bien, el hecho de que existan posiciones que concentren más poder puede no ser evidente. Este mismo Desarrollador relata cuándo y cómo esta situación resultó más clara, cuando Canonical contrató a Desarrolladores de Debian para Desarrollar Ubuntu,[24] una distribución derivada de Debian:

> **micah:** The power that these different groups have had kind of became more apparent, at least to me, when most of this key infrastructure positions ... these people were hired to work by Ubuntu, Canonical. And all of their time was eaten by this, and they had no time for Debian. So they were still in these positions, and were not doing anything, and there was no clear way to put someone else in those positions, or remove them. [...] The Debian security team had this problem, the system administrators had this problem, FTP Masters have had this problem [...] Because these people won't leave, and you can't do something to go around them. Then they are controlling things and that makes it obvious they have this power. [...]
>
> It was not deliberate, you know, "we are going to sabotage Debian", it was just they're working too hard and didn't have the time. So I don't

[24] Este momento aparece en muchas de las entrevistas realizadas como el momento más crítico respecto a los problemas de la acumulación de poder, personalizados en la figura de James Troup (*elmo*), que con su control del archivo y del proceso de creación de nuevas cuentas en Debian contribuía al sentimiento de bloqueo de muchos miembros de Debian.

> think any of these people were trying to be malicious, they just became overwhelmed. And there were no clear ways to replace them, or a clear way to add people, or they would not respond to people that wanted to be added. So it became kind of a problem because there were a lot of these positions where this was happening at the same time. So I think now has changed a little bit, but it could be better.

Y nos encontramos aquí de nuevo con que esa situación de concentración de poder que entra en tensión con la doocracia en tanto que puede dificultar que los Desarrolladores realicen su trabajo, es producto de esa misma doocracia:

> **zack:** In Debian, like in most free software projects, who does the work is actually the one who gets to decide. So, when some people who have being doing a lot of work, and thanks to that, they earn some important places which are key in Debian [...] you can argue that power is concentrated in these people. [...]
>
> We have never had any principle, or any guideline, or any de facto practice, of like saying, you are doing too much, we don't have anything like that. But when it comes to these positions which are core elements, one can say that someone is doing too much, that is possible.

3.2.4.1 FTP Masters y DAM.

Veamos brevemente dos de estas posiciones que controlan el acceso. El equipo *FTP Master* es el encargado de mantener el archivo en el que se encuentran los paquetes ofrecidos por Debian, y de aceptar o no los nuevos paquetes, según criterios de calidad y de conformidad de sus licencias con las *Debian Free Software Guidelines*.[25] Tiene entonces una gran capacidad de decisión sobre qué *software* forma parte de Debian y sobre sus actualizaciones:[26]

[25] Véanse las secciones 5.2.3, *Las licencias en Debian* y 5.3.2, *Organizar la colaboración ¿Existe la obligación de recibir en Debian?*

[26] Una buena descripción de las tareas del equipo se encuentra en Debian-devel-announce (2010a), donde un FTP Master solicita voluntarios para el equipo.

> The ftpmasters team wields a considerable amount of power over what does, and doesn't, make it into Debian package archive. For the vast majority of packages, the decisions may be cut and dried. At least there are relatively clear-cut guidelines based on known license problems, lack of licensing information, a failure for packages to build from source, policy violations, or any number of other known issues.
>
> The ftpmasters also have room for discretion in applying the rules and may reject packages for other reasons. Consider, for example, the decision to reject qmail packages from inclusion. This was less about Debian Policy and more, apparently, about the ftpteam's opinion of Qmail (Brockmeier, 2010).

La otra posición es la de DAM (Debian Account Manager). Así se definen sus funciones (Wiki.debian.org, s.f.[n]):

> The Debian Account Managers (DAM) are responsible for maintaining the list of members of the Debian Project, also known as Debian Developers. DAMs are authoritative in deciding who is a member of the Debian Project and can take subsequent actions such as approving and expelling Project members.

Una cuestión señalada repetidamente en numerosas entrevistas durante el trabajo de campo es que a esta concentración de poder en determinadas posiciones hay que añadirle el hecho de que algunas de ellas están ocupadas por los mismos individuos.[27] En concreto, la posición de DAM y FTP Master. Por ejemplo:

> **tincho:** Cuando yo estaba entrando, [...] en ese momento había mucho sentimiento de cabal en debian. Era el secreto a voces de que Debian está manejado por un cabal y que todo los que no estamos en el cabal estamos enojados con eso porque el cabal decide cosas de puertas adentro que nos afectan a todos y no aceptan cosas de afuera. [...]

[27] Sobre estos dos equipos y esta cuestión, véase también la sección 6.1, *De la comunidad al colectivo sociotécnico*.

> El cabal en ese momento que más afectaba quizás era el FTP Master, DAM, [...] son los dos más importantes y con los que más la gente estaba enojada [...] El FTP Master básicamente decide qué es Debian, decide qué entra y que no entra en Debian, y cómo se mueven los paquetes de un lado para otro, y eso nos afecta a todos. [...]
>
> Con DAM yo creo que era más chocante aún porque estaba todo ese sentimiento de, bueno, hay gente queriendo entrar y se tiene que pasar dos años esperando porque esta gente hace esto y no deja entrar a más nadie. Porque claro el problema no es que exista el cabal, sino que no está funcionando bien, ese es el problema. Porque claro, siempre fue igual, pero mientras fue eficiente, bueno, que hagan su trabajo. Lo hacen a oscuras y entre ellos. Bueno, no me importa mientras lo hagan bien. Pero cuando empezó a haber problemas porque la cola de NEW tardaba mucho en vaciarse, y entonces tenías que esperar un montón. Siempre está el problema de que no te aceptan un paquete, y no sabes por qué, [...] Ahí la gente decía, yo quiero ayudar, pero no me dejan. [...] Pero claro se daban las dos situaciones combinadas. Era un grupo cerrado, que no dejaba entrar a nadie, y encima era ineficiente. Resultado lógico: la gente quiere cambio.

Esto era así sobre todo antes del inicio del trabajo de campo. Sin embargo, todos los entrevistados señalaban también que había un esfuerzo por añadir más gente a los equipos centrales, y en la actualidad hay más diversidad en ellos. Por otra parte, sigue habiendo una barrera para entrar en esos equipos debido al nivel de exigencia de tiempo y dedicación que supone pertenecer a ellos, así como a la diversidad de tareas especializadas que suponen. Además, dada la especialización y exigencia técnica de estos equipos, la enseñanza para realizar ese trabajo sólo pueden realizarla ellos mismos, por lo que *de facto* controlan ellos mismos la entrada al equipo. Pero en las entrevistas realizadas, numerosos Desarrolladores manifestaban que a partir del propio esfuerzo y trabajo siempre era posible involucrarse y entrar en ellos.

3.2.4.2 *There Is No Cabal.*

Medio en broma medio en serio, son recurrentes las bromas entre los Desarrolladores sobre la existencia de una *cabal* (camarilla o grupo de conspiradores) en Debian, en numerosas ocasiones usando la frase «There Is No Cabal (TINC)» (Coleman, 2013, págs. 136-137),[28] con intención irónica. Se alude así precisamente a grupos que concentran más capacidad de decisión y son capaces de imponer su voluntad respecto a la marcha general del proyecto. También a grupos de trabajo en los que es difícil entrar si no se tienen las relaciones personales adecuadas. Curiosamente, se suele usar el término acompañado del correspondiente gentilicio: *the German cabal*, *the English cabal* o *the Spanish cabal*, entre otros. Algunos son más irónicos que otros. En el uso de este término encontramos simbolizada la tensión entre la estructura horizontal, no jerárquica, a la que se aspira, y la aparición de jerarquías y posiciones con más poder de decisión. Por ejemplo:

> **dato:** Lamentablemente es un paradigma que ha ocurrido muchas veces en Debian, el hecho de que muchos equipos, sobre todo equipos de lo que llamamos la infraestructura más *core*, más central al proyecto, de la que depende el buen funcionamiento de todo, han sido muchas veces como agujeros negros, en los que entraban las peticiones, o las preguntas, o lo que fuera, y de ahí nunca salía nada. Es lo que a veces se llamaba la vieja cábala, la *old cabal*, la gente que estaba en esos puestos de responsabilidad, pues siempre pasaba de todo el mundo y ahí se las dieran todas. Afortunadamente, poco a poco, esa gente ha ido siendo reemplazada.

La figura de la *cabal* está entonces en contradicción con la apertura propia de Debian, de ahí la desconfianza ante las mismas. Además, impiden el establecimiento de conexiones que hemos visto, y seguiremos viendo, es una de las características fundamentales del trabajo técnico que se hace en Debian. A pesar de la similitud en el nombre, esos agujeros negros son lo contrario de las cajas negras que hemos visto que ayudan a articular Debian (véase la sección 2.2.6, *De las cajas negras a los parches como metáfora de lo técnico*).

[28] El uso del término y de esta frase se remonta a Usenet y la *Backbone Cabal*. Sobre esta cuestión puede consultarse *The Jargon File*, (Raymond y Steele, 2004).

Volveremos a encontrar estas tensiones al considerar los procesos de reforma del proceso de ingreso en Debian.

3.2.5 Doocracia y agencia: la mediación técnica.

En la doocracia como principio organizativo y de toma de decisiones volvemos a encontrar incorporada una determinada concepción y construcción de la diferencia entre lo técnico y lo social (véase la sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*). Se tiene capacidad de decisión individual, como Desarrollador, sobre cuestiones técnicas, sobre aquellas cuestiones y elementos sobre los que se puede ofrecer una solución técnica. Se tiene poder de decisión en tanto se es capaz de ofrecer una forma de mediación técnica; en tanto se es capaz de inscribir un programa de acción en un dispositivo técnico, como veíamos a propósito del *bug* de `python` y la controversia sobre la inclusión de *firmware* no libre. O al menos, son susceptibles de ser decididos individualmente aquellos asuntos que se pueden resolver en relación a los privilegios técnicos acumulados y al acceso privilegiado a las máquinas y los servicios.

Veamos un caso para ilustrar concisamente la cuestión. Es un ejemplo sencillo que sirve para poner de manifiesto algunos mecanismos propios de la práctica técnica en Debian. Tradicionalmente, el entorno de escritorio (la interfaz gráfica que permite al usuario manejar su ordenador de manera intuitiva y gráfica, como si fuera un escritorio) que se instalaba en Debian de manera predeterminada ha sido GNOME (GNOME, s.f.). Es posible y en realidad muy sencillo cambiar a otro, pero éste es el que se instala durante el proceso de instalación si el usuario no elige explícitamente otro, por lo que de hecho es el que muchos nuevos usuarios acaban usando. El programa que en Debian realiza esta elección (normalmente durante la instalación del sistema operativo) es `tasksel`, una utilidad para seleccionar e instalar tareas (*tasks*), conjuntos de programas que al ser instalados permiten que el ordenador realice determinadas funciones. Algunos ejemplos son *desktop*, *web-server*, *database-server*, *laptop*, o *manual package selection* para elegir los paquetes a instalar. Escoger la tarea *desktop* instala GNOME, si se elige *manual package selection* se puede escoger otro.

Pues bien, en noviembre de 2013 se cambió esta elección para que el en-

torno de escritorio predeterminado al marcar la tarea *desktop* fuese Xfce (Xfce, s.f.), mucho más ligero, en lugar de GNOME. En Agosto de 2014, un Desarrollador de Debian pidió en un *post* en su *blog* titulado «A pile of reasons why GNOME should be Debian jessie's default desktop environment» (Mallach, 2014), y en un mensaje de correo con el *subject* «Reverting to GNOME for jessie's default desktop» a las listas `debian-project`, `debian-devel`, `debian-boot`, `pkg-xfce-devel`, `debian-gtk-gnome`, `debian-accessibility`, `debian-cd` y `tasksel`,[29] que se volviese a adoptar GNOME como la elección por defecto en la próxima versión del sistema operativo, de nombre *Jessie*. Se pidió en este momento al estar cercano el «congelamiento» (*freeze*) de los paquetes que formarán parte de esta versión. Una vez pasado ese momento, no se pueden en principio introducir cambios de ese tipo en los paquetes que formarán parte de esa versión de la distribución.

El contenido de esta cuestión y de la discusión que siguió es interesante, con argumentos en favor de una u otra opción. Pero lo que aquí nos interesa es sobre todo cómo se realizó el cambio. Dice este Desarrollador al final de su post, tras ofrecer las razones por las que que GNOME sería una mejor alternativa:

> In short, we think defaulting to GNOME is the best option for the Debian release [...] We believe tasksel should again revert the change and be uploaded as soon as possible, in order to get people testing images with GNOME the sooner the better, with the freeze only two months away.
>
> We would also like that in the future, changes of this nature will not be announced in a git commit log, but widely discussed in debian-project and the other usual development/decision channels, like the change of init system happened recently. We will, whichever the final decision is, continue to package GNOME with great care to ensure our users get the best possible desktop experience Debian can offer.

Lo relevante es que el cambio en una opción con consecuencias amplias ha

[29] Pongo todas las listas para mostrar los equipos implicados en principio en el cambio. La discusión, por ejemplo en la lista `debian-devel`, puede seguirse en Debian-devel (2014a) y los mensajes siguientes.

sido realizado por `tasksel` (y por lo tanto por sus mantenedores), y anunciado al proyecto en el *log* de *commits* del sistema de control de versiones en el que se mantiene el paquete, en este caso sobre `git`. Se explicarán estos términos al hablar del papel de los Sistemas de Control de Versiones, en la sección 4.8, *Repositorios de software libre*. De momento basta decir que un *commit* es un conjunto de cambios a una serie de archivos que en ese momento se hacen disponibles a quien tenga acceso al sistema de control de versiones en el que se producen, y el *log* es el registro de ese cambio. Parte del *log* del *commit* en el que se realiza el cambio es el siguiente:[30]

```
author    Joey Hess <joey@kitenet.net>
           Sun, 3 Nov 2013 20:41:25 +0200 (14:41 -0400)
committer  Joey Hess <joey@kitenet.net>
           Sun, 3 Nov 2013 20:42:09 +0200 (14:42 -0400)
commit    dfca406eb694e0ac00ea04b12fc912237e01c9b5
tree    081df6f28b0034628493845cd01f7a9edacd0091
parent    4d76118a1d97bcbf172e678033415a0451ee9d8d

Change default desktop to xfce.
This will be re-evaluated before jessie is frozen. The evaluation will
start around the point of DebConf (August 2014). If at that point gnome
looks like a better choice, it'll go back as the default.
Some criteria for that choice will include:
[...]
--
Hello to all the tech journalists out there. This is pretty boring.
Why don't you write a story about monads instead?
```

Y el parche que muestra los cambios introducidos por este commit:

```
 debian/changelog | 8 ++++++++
 debian/control   | 2 +-
```

[30] En Salsa.debian.org (2013a). La firma es una muestra del humor tan presente siempre en Debian.

```
 debian/templates | 2 +-
 3 files changed, 10 insertions(+), 2 deletions(-)

diff --git a/debian/control b/debian/control
index 919a6e7..d4d2dc9 100644
--- a/debian/control
+++ b/debian/control
@@ -45,7 +45,7 @@ Recommends:
 # The order here is significant when installing this task manually;
 # when tasksel installs this task it instead selects one of these based
 # on the tasksel/desktop debconf setting.
-    task-gnome-desktop | task-kde-desktop | task-lxde-desktop | task-xfce-desktop,
+    task-xfce-desktop | task-gnome-desktop | task-kde-desktop | task-lxde-desktop,
 # For use by third-party apps.
     xdg-utils,
 # mdns/zeroconf stuff

diff --git a/debian/templates b/debian/templates
index 423aa87..24fe044 100644
--- a/debian/templates
+++ b/debian/templates
@@ -18,7 +18,7 @@ _Description: Choose software to install:
 Template: tasksel/desktop
 Type: multiselect
 Choices: gnome, kde, xfce, lxde
-Default: gnome
+Default: xfce
 Description: The desktop environment to install when the desktop task is selected
  This can be preseeded to change the default.
```

Las primeras líneas nos indican que se han producido cambios en tres archivos, y el número de líneas cambiadas en cada uno de ellos. A continuación se muestran los cambios realizados a los dos ficheros que implementan efectivamente el cambio. Las líneas que empiezan con - han sido suprimidas, las que empiezan

con + han sido añadidas. La línea con @@ indica dónde se produce el cambio, y el resto se muestra para dar contexto. Vemos así claramente en qué consiste el cambio y dónde se ha producido.

Aunque esta es una decisión que afecta al Proyecto en su conjunto y a sus usuarios, quien de hecho está capacitado para tomarla son los mantenedores del paquete `tasksel`, que tienen acceso a cambiar su comportamiento. A ellos se dirige la petición de que GNOME vuelva a ser el escritorio predeterminado. El cambio toma la forma de un parche, y se produce a través de esta mediación técnica (sobre el parche como forma privilegiada de mediación técnica, véase la sección 2.2.6, *De las cajas negras a los parches como metáfora de lo técnico*). Vemos entonces cómo toma forma la doocracia: el encargado de decidir si y cómo se realiza algo es quien efectivamente lo lleva a cabo, teniendo suficientes privilegios de acceso para ello.

En realidad, el mantenedor de `tasksel` que implementó el cambio indica en la discusión que el cambió se discutió en la DebConf (la conferencia que cada año reúne a los miembros de Debian) del año anterior, negando que fuese anunciada sólo en un *commit* (Debian-devel, 2014b):

> Incidentially, I don't much appreciate the counterproductive sniping that Jordi added in his blog post about this. Perhaps you're not aware, Jordi, but switching to xfce was discussed at last DebConf. It was not done "announced in a git commit log".

Pero también es cierto que la decisión final pasa necesariamente por ellos, que se han convertido en lo que Callon (1984) llama punto de paso obligatorio (véase la sección 2.1.7, *Procesos de traducción*). Así, otro Desarrollador de Debian (y miembro del Comité Técnico) interviene en la discusión (Debian-devel, 2014c):

> Fascinating as this discussion is, I think it is at risk of becoming too much of a time sink. I think that it would be useful to have some authoritative guidance from those in Debian who are responsible for this decision. AFAICT that is the tasksel maintainers.

> So I would appreciate it if the tasksel maintainers would let us know:
>
> Do you intend to review (or are you reviewing) the decision taken in July 2012 [1] ? If so, is this discussion here on -devel useful ? If it is useful, what questions should we be focusing on ?
>
> Ian.

Aquí vemos explícitamente la atribución de autoridad en este cambio. La discusión puede contribuir mucho al resultado, pero quien tiene al final la última palabra son los mantenedores de `tasksel`, incluso aunque en ninguna parte se les delegue explícitamente la decisión sobre cuál ha de ser el entorno de escritorio predeterminado. La cuestión de quién puede decidir no llega a plantearse en forma de controversia (se acepta que pueden decidirlo, y se les pide que tomen una determinada decisión), dado que desde el principio aparece como una decisión técnica, al residir el cambio en un parche a una pieza de *software* determinada. Y eso que gran parte de la discusión subsiguiente sobre qué entorno de escritorio es más adecuado se centraba en cuestiones que los miembros de Debian suelen incluir como sociales: la accesibilidad del entorno operativo (especialmente para ciegos), el ancho de banda necesario para descargar e instalar los diferentes entornos (consideración que en muchas partes del mundo es relevante), o la popularidad de las diferentes opciones. Pero al ser una decisión técnica, como vimos en la sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*, la decisión corresponde al mantenedor. Aunque ciertamente sí existen en el Proyecto Debian mecanismos para corregir decisiones que son producto del mantenimiento de un paquete. La decisión en este caso se deja en manos de los mantenedores de `tasksel`, pero siempre se podría pedir una resolución del Comité Técnico o incluso una votación a través de una Resolución General.

En este caso, lo que hizo el mantenedor de `tasksel` fue abrir una página (Wiki.debian.org, s.f.[o]) en la *wiki* de Debian unos días después, donde diferentes equipos daban argumentos, desde su área de responsabilidad, sobre la idoneidad de uno u otro entorno de escritorio como entorno por defecto. El objetivo de esta página era: «to collect information about the status of desktop environments in Debian, to aid the tasksel maintainers in deciding which desktops are suitable candidates to be the default desktop». Recogida la información, en septiembre de

2014 se volvió a GNOME como escritorio por defecto. La justificación del cambio se puede leer en el *log* del *commit* correspondiente (Salsa.debian.org, 2014).

La doocracia aparece entonces como una forma de crear asimetrías y poder a través de la creación de cajas negras (o, en sentido general, de parches). Siguiendo a Callon y Latour (Callon y Latour, 1981), consideremos qué es lo que hace a un actor tener más o menos poder, capacidad de decisión. Es decir, convertirse en un macro-actor. Estos no son *a priori* mayores (con más poder) que los demás. El tamaño es siempre consecuencia del hecho de que los *actores* son también *redes*, conjuntos de asociaciones:

> A difference in relative size is obtained when a micro-actor can, in addition to enlisting bodies, also enlist the greatest number of durable materials.
>
> [...]
>
> By associating materials of different durability, [...] Only thus can one 'grow'. [...] it is necessary to enrol a *little more* than relationships, alliances and friendships. An actor grows with the number of relations he or she can put, as we say, in black boxes (Callon y Latour, 1981, pág. 284).

Son las asociaciones que un individuo establece con los dispositivos tecnológicos las que en definitiva le proporcionan poder o capacidad de decisión. La doocracia funciona porque existe mediación técnica en las prácticas relacionadas, se decide de acuerdo con la doocracia en prácticas mediadas técnicamente. Y no hay otra manera de establecer esas asociaciones que efectivamente trabajando para unir diferentes elementos en una configuración estable (por ejemplo, creando un paquete o un servicio nuevo en las máquinas de Debian). Este es el sentido que recoge en definitiva el uso del concepto de doocracia. Y por eso los dispositivos técnicos son tan importantes para entender una realidad como el Proyecto Debian como un colectivo, un entramado sociotécnico. Sin ellos no habría colectivo duradero. Tampoco son los únicos. Tan importantes son las asociaciones y relaciones entre individuos como entre estos y los artefactos técnicos:

> We end up with actors of different size even though they are all isomorphic, because some have been able to put into black boxes more elements durably to alter their relative size. [...] directing our attention not to the social but towards the processes by which an actor creates lasting asymmetries. That among these processes some lead to associations which are sometimes called 'social' (associations of bodies), and that some of the others are sometimes called 'technical' (associations of materials), need *not* concern us further (Callon y Latour, 1981, pág. 285).

Hay que tener en cuenta que en este proceso surgen actores nuevos. En el caso del mantenimiento de un paquete, el Desarrollador y aquellos elementos con los que se asocia (co-mantenedores, pero también el código que empaqueta, los repositorios que lo almacenan, o el BTS, por ejemplo). Recordemos los ejemplos que hemos ido viendo en este capítulo. Los actores relevantes en la discusión sobre `python`, o sobre el *firmware*, no eran solamente los individuos implicados considerados aisladamente. Entender su acción, su tamaño relativo, su capacidad de decisión diferencial, exige entenderlos a partir de las asociaciones que mantienen con esos otros elementos. Es decir, como actores-redes. En el contexto de la descripción, su papel en los procesos de acción social es inseparable de los elementos heterogéneos con los que se asocian:

> macro-actors are micro-actors seated on top of many (leaky) black boxes.
>
> [...]
>
> What is an 'actor'? Any element which bends space around itself, makes other elements dependent upon itself and translates their will into a language of its own. An actor makes changes in the set of elements and concepts habitually used to describe the social and the natural worlds (Callon y Latour, 1981, pág. 286).

Teniendo en cuenta además, como vimos antes, que en muchas ocasiones quien establece esas asociaciones es un equipo de trabajo y no un individuo parti-

cular. Buena parte de lo que se hace en Debian es realizado por equipos en los que colaboran diferentes personas.

En definitiva, la doocracia, además de como una forma de gobierno de ciertas comunidades, se ha de entender también como una forma de constituir actores de una determinada manera, que necesita la mediación de sistemas técnicos que lo permitan. En tanto que constituyen actores y permiten su inclusión, estos sistemas han de considerarse como *mediadores* (véase la sección 4.2.2, *Mediadores e intermediarios. Las figuras del* bug *y del* hacker). Al mismo tiempo, la doocracia hace más difícil que un participante en estos colectivos sea un simple *intermediario* de los deseos u objetivos de otro. Ambos elementos nos ayudan a repensar cómo se distribuye y transforma la agencia en este contexto.

Lo que nos muestra la concepción de la doocracia que hemos visto es entonces una determinada concepción de la agencia colectiva en la que ésta es producto de la agencia de cada uno de sus miembros, pero de tal modo que ésta está siempre mediada por dispositivos tecnológicos. Si la acción de un individuo afecta al colectivo o a otros desarrolladores, es porque es una acción en relación a dispositivos y artefactos técnicos que los ponen en relación, que median entre ellos. La doocracia no se puede entender sin la relación, la asociación, con los dispositivos técnicos. Supone la capacidad de reclutar apoyos y aliados, entre los cuales están también los elementos no humanos del colectivo. Y por esto precisamente se produce esa cierta concentración y acumulación de poder de decisión que hemos visto. Lo que ésta nos indica es que se ha establecido una asociación más fuerte, más estable, con esos dispositivos. Que se han construido cajas negras de mayor tamaño, con más asociaciones de elementos en su interior. Que se han movilizado más recursos.

La doocracia es una forma de repartir agencia y poder en Debian, una forma convencional de las relaciones sociales que se dan en el proyecto. Dar cuenta de ella exige el uso de una concepción semiótica de la agencia, como la que hemos seguido a partir de Kockelman, y de Callon y Latour. Puesto que determinar en la práctica social cuando es o no aplicable, decidir quién puede resolver una cuestión, depende de que el asunto en cuestión sea definido o resignificado como técnico o social.

Pero la doocracia no podría haber surgido sin una manera concreta de organizar el trabajo de los Desarrolladores en la creación y mantenimiento de la distribución, una manera de empaquetar. Una manera que surge no para dar poder a individuos, sino para hacer sostenible y viable el proyecto, reduciendo las necesidades de coordinación y acuerdo previos, y facilitando la incorporación de nuevas contribuciones. Veremos esto en el capítulo 5, *El don del* software *libre*. El surgimiento y extensión de la idea de la doocracia como forma convencional de orientar las acciones y las prácticas es un reconocimiento de que ésa ha sido de hecho, más que la meritocracia, la forma de funcionar. Esto no ha sido considerado algo positivo en todos los casos, ahí están las quejas sobre las cábalas y la concentración de poder por tener determinados privilegios de acceso. También las tensiones que causa la existencia de derechos adquiridos sobre los paquetes que se mantienen.

En la doocracia como forma de toma de decisiones se manifiestan entonces toda una serie de tensiones que atraviesan la constitución de Debian como colectivo sociotécnico, y que son fundamentales para entender su dinámica social. Es decir, para entender las asociaciones y formas de relación que se producen entre los diferentes individuos y entre éstos y los artefactos técnicos que los constituye como colectivo. En las secciones y capítulos siguientes nos seguiremos preguntando por esas tensiones, así como por algunos de los requisitos para que pueda funcionar la doocracia y que le dan sentido. Entre ellos, la existencia de medios y protocolos abiertos (sección 3.3, *Debian como público recursivo*), el reconocimiento de la obligación de recibir como elemento integrante del don (sección 5.3, *La obligación de recibir*), la mediación tecnológica (capítulo 4, *El carácter híbrido del* software. Bugs, hackers, *paquetes y repositorios*) o el establecimiento de relaciones de confianza (sección 6.4, *La construcción tecnosocial de la confianza*).

3.3 Debian como público recursivo.

Junto al de doocracia, hay otro concepto que considero esencial para dar cuenta de la forma de organización del Proyecto Debian, el de *público recursivo*, acuñado por el antropólogo Chris Kelty. En línea con lo que venimos estableciendo, Kelty insiste en la idea de que un estudio de la cultura del *software* libre supone la consideración de las múltiples relaciones que se producen entre personas y ob-

jetos tecnológicos, en una gran cantidad de lugares: «Free software, as a cultural practice, weaves together a surprising range of places, objects, and people» (Kelty, 2008, pág. 2). El constituir un público recursivo es uno de los elementos que da consistencia a esta trama que estamos describiendo.

Un público recursivo es un público compuesto por usuarios dotados de diferentes grados de conocimiento tecnológico, e implicados activamente tanto en usar las tecnologías como en crear y mantener una infraestructura compartida y en sostener sus reglas de uso (Kelty, 2005a, 2008). Éste existe no sólo porque se crea y mantiene una infraestructura, sino porque produce y sostiene una dimensión de reflexividad en sus discursos y prácticas. Esta dimensión se encuentra en la reflexión sobre las tecnologías utilizadas, pero también sobre las normas que la regulan. Así lo define Kelty:

> A recursive public is *a public that is vitally concerned with the material and practical maintenance and modification of the technical, legal, practical, and conceptual means of its own existence as a public; it is a collective independent of other forms of constituted power and is capable of speaking to existing forms of power through the production of actually existing alternatives* (Kelty, 2008, pág. 3).

Aquí la independencia de otras formas de poder constituido no ha de tomarse absolutamente, pero cobra un mayor sentido al tener en cuenta la independencia de Debian respecto a cualquier empresa. Pero el elemento fundamental de este concepto es la «radical technological modifiability of their own terms of existence» (Kelty, 2008, pág. 3). Esta característica es claramente aplicable al Proyecto Debian, especialmente si tenemos en cuenta que

> Recursive publics are "recursive" not only because of the "self-grounding" of commitments and identities but also because they are concerned with the depth or strata of this self-grounding: the layers of technical and legal infraestructure which are necessary for, say, the Internet to exist as the infrastructure of a public (Kelty, 2008, pág. 8).

Y efectivamente, el Proyecto Debian no sólo crea una distribución de *software*. También desarrolla y mantiene por sí mismo toda la infraestructura necesaria para que esto se pueda llevar a cabo. Administra sus propias máquinas, desarrolla *software* para el mantenimiento de la distribución, revisa minuciosamente las licencias de todo paquete de *software* que distribuye, administra sus propias listas de correo y su red de servidores de IRC, organiza sus Conferencias Internacionales ... Como escribe un Desarrollador de Debian en su *blog* (Zacchiroli, 2010g),[31] esto es fundamental para la «cultura de la libertad» que caracteriza al Proyecto:

> **The culture of freedom lies in the details**
>
> Here is an interesting blog post by Bradley Kuhn about Ubuntu, Debian, and (warning: my interpretation ahead) the culture of freedom.
>
> While reading it, I had kinda moment of truth, because just yesterday I was musing with Mehdi and Lucas on the fact that Debian is basically the only remaining distribution among the mainstream ones (if that means something) that is **free from the ground up**, including its infrastructure. We «just» seek hardware via donations and then we run, thanks to the amazing work of DSA, our own **free infrastructure** on top of it.
>
> Let's cherish this value!
>
> [...]

Aquí aparecen dos características fundamentales de los públicos recursivos. Por una parte, que la infraestructura sea **propia**, desarrollada y mantenida por el propio Proyecto. Por otra parte, e igualmente importante, que ésta sea, además, **libre**, construida con *software* libre y no dependiente de servicios externos, comerciales.[32] Ambas características no son independientes. Que sea realmente propia y controlada por la comunidad Debian requiere también que sea libre, no propietaria. Que sea libre implica que no existe control externo sobre su funcionamiento. Como repiten en numerosas ocasiones los miembros de Debian, una de las características que distingue a esta distribución de otras es que se trata, en

[31] El *post* al que se refiere se puede encontrar en Kuhn (2010).
[32] Sobre la importancia de que este *software* sea libre, véase Hill (2010).

toda su extensión, de una *community-driven distro*, una distribución mantenida por la comunidad, independiente de cualquier empresa.

Debian usa la distribución Debian en sus servidores para desarrollar esa misma distribución. En tanto distribución se configura entonces como público recursivo. A esto se añade el hecho de que son los Desarrolladores de Debian ante todo los que se consideran usuarios a los que va dirigida la distribución. En realidad, esto es algo típico en el entorno del *software* libre. Eric Raymond expresa así la primera lección en el desarrollo de *software* libre, en una expresión muy repetida: «Every good work of software starts by scratching a developer's personal itch» (Raymond, 1999, pág. 31).

La idea que se expresa aquí es que el *software* desarrollado como *software* libre suele responder a una necesidad del propio Desarrollador, más que a las de usuarios externos. Los usuarios que se tienen en cuenta al diseñar un programa suelen ser los mismos Desarrolladores. Al mismo tiempo, eso facilita que los usuarios se conviertan en desarrolladores. Esto ocurre así en el *software* libre, como señala Ratto (2003) indicando que Linux está diseñado para ser rediseñado por otros desarrolladores y usuarios. Y desde luego en Debian. Un Desarrollador de Debian resumía así una pequeña investigación sobre los motivos y el proceso que llevó a nuevos participantes a contribuir en el desarrollo de Debian, y que se refleja también en muchas de las entrevistas realizadas para la presente tesis: «if I try to draw a typical picture of the new contributor, he/she is a long time Debian user who started contributing to «scratch an itch»: package something s/he developed, depend on in its day job, etc» (Debian-project, 2013).

Y es que un elemento propio de Debian, y del *software* libre en general, es la imposibilidad de separar nítidamente usuarios y desarrolladores. Ambos grupos se solapan en gran medida. Veamos un mensaje a una lista de correo que resume muy claramente esta cuestión (Debian-devel, 2016b):

```
To: debian-devel@lists.debian.org
Subject: Re: Debian does not have customers, but users
From: Jonas Smedegaard <dr@jones.dk>
Date: Sun, 18 Sep 2016 15:27:48 +0200
```

```
Quoting The Wanderer (2016-09-18 15:02:25)
> On 2016-09-18 at 08:40, Santiago Vila wrote:
>
> > On Sun, Sep 18, 2016 at 02:00:26PM +0200, Abou Al Montacir wrote:
> >
> > > you will end being a community of geeks developing SW for
> > > themselves only.
> >
> > Debian is a volunteer project made by its users.
> >
> > So we are already a community developing SW for ourselves, the
> > users, and there is nothing wrong with that.
> >
> > Once you realize that «people making Debian» and «its users» are
> > really the same thing, you will see why this discussion does not
> > make much sense.
>
> One nitpick:
>
> There are many, many people who are users of Debian who do not
> contribute to the development of Debian, except possibly by way of
> filing bug reports.

That exception is the whole point.

   • Jonas

[x] quote me freely [ ] ask before reusing [ ] keep private
```

Desde luego, y en primer lugar, los Desarrolladores son usuarios del sistema. Pero también, en buena medida, los usuarios tienen muy fácil convertirse en Desarrolladores, aunque sólo sea informando de fallos. En definitiva, «“people making Debian” and “its users” are really the same thing».

3.3.1 La infraestructura de Debian.

O, como escribe en su *blog* un Desarrollador de Debian, «Debian is eating its own dog food more than ever» (Hertzog, 2011a):

> Eating its own dog food is very important if you want to build a Linux distribution and claim with some confidence that it’s of quality and usable.
>
> Debian does quite well nowadays in that respect.

> There were times where the mailing list server was using Qmail (non-free at that time, and thus not part of Debian) but that's long gone. We have also seen our build infrastructure relying on software that was not public and not packaged in Debian, but that also is history.
>
> The Debian System Administration team (DSA) maintains more than 140 servers running Debian. [...]
>
> PS: If you want to learn more about the setup that the DSA team uses, head to dsa.debian.org. You'll find all their repositories and some of their internal documentation.

Debian System Administrators o DSA (Dsa.debian.org, s.f.) es el equipo que mantiene la infraestructura de Debian. La lista de máquinas que mantiene se puede ver en Db.debian.org (s.f.[a]), con una descripción de cada una de ellas. Su uso está regulado por la «Política de Uso de Máquinas de Debian» (Debian.org, s.f.[p]), que todos los miembros de Debian deben firmar. El Proyecto Debian mantiene además sus propios servicios para el funcionamiento de la distribución, y desarrolla algunas herramientas propias para su funcionamiento, ofreciendo sus propios servicios a los Desarrolladores (Wiki.debian.org, s.f.[p]). En la página de su *wiki* que lista las ventajas que los desarrolladores de *software* libre pueden encontrar al desarrollar para Debian se dice lo siguiente (Wiki.debian.org, s.f.[q]):

> **Project infrastructure**
>
> The Debian Project has scripts, remote services and resources for automatic processing to the maximum extent for almost every item of Debian Policy or aspect of software development.
>
> You will be almost free of repetitive work, freeing your time to develop technically solid solutions.

Algunos de estos servicios y herramientas, sin ánimo de ser ni mucho menos exhaustivo, incluyen por ejemplo:

Devotee (*DEbian VOTe Engine*), el sistema usado en las votaciones del Proyecto Debian (Debian.org/vote, s.f.).

DAK (*Debian Archive Kit*), los *scripts* usado para mantener el archivo del Proyecto Debian (Ftp-master.debian.org, s.f.[a]).

Cdbs y debhelper Herramientas para ayudar al empaquetado. Véase la sección 2.1, *Bug #573745*.

BTS El *Bug Tracking System*. Véase el cuadro 2.3, *Bug Tracking System*.

PTS El *Package Tracking System*. Para encontrar información sobre los paquetes que hay en Debian (Packages.qa.debian.org, s.f.).

Listas de correo Véase el cuadro 2.1, *Listas de Correo*.

OFTC La *Open and Free Technology Community*, la red de IRC (véase el cuadro 1.1, *IRC*) donde se alojan los canales del Proyecto Debian. Es una organización ligada a *Software in the Public Interest* y también a Debian.

Salsa Servidor (basado en GitLab) donde se alojan herramientas para el desarrollo y empaquetado colaborativo (Salsa.debian.org, s.f.[a]).

Y muchos otros.

3.3.2 Apertura y libertad.

Una característica fundamental de estas infraestructuras y del público recursivo que ayudan a configurar es su *apertura*. Ésta es uno de los cinco componentes del *software* libre que distingue Kelty (2008):

> Openness and open systems are key to understanding the practices of Free Software: the open-systems battles of the 1980s set the context for Free Software, leaving in their wake a partially articulated infrastructure of operating systems, networks, and markets that resulted from figuring out open systems. [...]
>
> This "infrastructure" is at once technical (protocols and standards and implementations) and moral (expressing ideas about the proper order

> and organization for commercial efforts to provide high-tech software, networks, and computing power) (Kelty, 2008).

La infraestructura es al mismo tiempo técnica y moral. También lo es por tanto la apertura que le caracteriza. Veamos en qué sentido. En su capítulo 5, «Conceiving Open Systems», Kelty (2008) ofrece una interesante historia de este desarrollo, y de su importancia para el desarrollo no sólo del *software* libre sino también de las redes abiertas. Lo que puede ser más o menos abierto en estos casos son los protocolos y los estándares. DeNardis (2009) insiste también en este aspecto de la apertura, fundamentalmente en relación al gobierno de Internet: «The ongoing global spread of culture and ideas on the Internet [...] depends on the availability of open technical protocols on which technological universality and the pace of innovation and access is predicated» (DeNardis, 2009, pág. 1).

Y como Kelty, enfatiza que esta apertura ha de entenderse no sólo en un sentido técnico sino también moral y político:

> [...] protocols are political. They control the global flow of information and make decisions that influence access to knowledge, civil liberties online, innovation policy, national economic competitiveness, national security, and which technology companies will succeed (DeNardis, 2009, pág. 6).

Así lo expresa Kelty:

> Thus, techniques and design principles that are used to create software or to implement networking protocols cannot be distinguished from ideas or principles of social and moral order for these informants (Kelty, 2005a, pág. 186).

En un capítulo con el significativo título de «Technical Architecture shapes Social Structure: an example from the real world», el Desarrollador de Debian

Daniel Kahn Gillmor trata también la relación entre diseño tecnológico y diseño social:

> When you use the Internet, most of your communications rely on many different computers co-operating with each other. The computers co-operate with each other because they have agreed beforehand on a protocol.
>
> The protocols we use for communication shape not just the communications themselves, but social and economic structures beyond them. [...] This article shows how choices in digital communications infrastructure can also have an effect on our social fabric by focusing on one small example out of many (Kahn Gillmor, 2007, pág. 55).

En concreto, examina los protocolos TLS y SSL, que permiten la comunicación segura en Internet entre un navegador y un servidor. El problema que ve el autor en estos protocolos tal y como están implementados consiste en que los certificados (según el estándar X.509) que permite el cifrado de la comunicación y la autentificación del servidor han de ser emitidos por una Autoridad de Certificación. Para que el sistema funcione sin más preocupaciones para un usuario común, estas Autoridades han de conseguir ser incluidas «de serie» en los navegadores más utilizados, que así «confían» en ellas. Esto produce que el control de la seguridad de estas conexiones se concentre en manos de unas pocas empresas. Frente a esto, el autor propone el uso y la extensión de protocolos y estándares ya en desarrollo. Sustituir el uso exclusivo de certificados X.509 por certificados OpenPGP[33] (que permiten múltiples firmas del certificado, eliminando así la necesidad del papel central de las Autoridades de Certificación), permitiría una arquitectura de las conexiones seguras más abierta y menos centralizada. Permitiría conexiones seguras controladas por los usuarios:

> More than just needing secure communications, we need secure communications without faceless, unaccountable, politically-fickle, mer-

[33] Sobre esto véase la sección 6.4.1, *Identidad y claves GPG*.

> cenary gatekeepers. We need to take control of our own communications by taking responsibility for them (Kahn Gillmor, 2007, pág. 64).

Pero más allá del ejemplo concreto del que trata, lo relevante aquí es la insistencia en tomar la responsabilidad de las comunicaciones que configuran un público por parte de ese mismo público. Es decir, su constitución como público recursivo. Y esto es posible a partir del desarrollo de *software* libre porque en este caso son los propios interesados en que este cambio tenga lugar los que pueden hacerlo:

> One of the reasons to focus on Free Software [...] is that we have an opportunity to contribute changes that we want to see. The big proprietary software makers may not share our agendas, or may actually be antagonistic (Kahn Gillmor, 2007, pág. 65).

Es decir, en buena medida es la doocracia, la capacidad de tomar decisiones que da el hecho de poder tomar iniciativas en la construcción de *software*, protocolos y estándares, lo que permite la configuración de un público recursivo que se define esencialmente por su apertura. La doocracia contribuye así a configurar una jerarquía horizontal, plana, si los dispositivos tecnológicos en los que se apoya son abiertos. Como vimos, la doocracia implica que se han de tener los suficientes derecho de acceso. Esos puntos de acceso que vimos en la sección 3.2.4, *Concentración de poder y los límites internos de la doocracia*, han de estar entonces abiertos para que funcione. Un Desarrollador expresa así la importancia de la libertad de acceso:

> **clint:** When I joined I made lots of mistakes. Mistakes can be corrected. I think that it is easier to learn, to become better, than to have someone to change their philosophy. I think that if we can afford to have people make mistakes ... a lot of people have the mentality that things should be tightly controlled and no one should have access unless necessary, but I think that discourages people. I think that if you give people the power, and they have the right motivation, and they

> don't wanna cause harm, they can make mistakes and then fix them, and you are better off than if they can't even try.
>
> [...]
>
> People should be free to help.

Como hemos visto siguiendo a estos tres autores (Kelty, DeNardis y Kahn Gillmor), el de público recursivo es un concepto (aunque sólo lo usa Kelty) que pone de manifiesto la inseparabilidad del diseño social respecto del diseño técnico. Para que haya una mayor libertad en los colectivos sociotécnicos, es necesario que haya apertura en los protocolos y las infraestructuras.[34] Esto no es algo exclusivo de la tecnología implicada en el diseño de *software* o de Internet, pero el hecho de que su ya relativa novedad, junto con la posibilidad de abrir las cajas negras (Internet se basa en un conjunto de protocolos y estándares abiertos, y el *software* del que estamos tratando aquí es *software* libre, abierto) que los constituyen lo hace más evidente.

Una última cita nos muestra, aun sin usar el concepto, la estrecha implicación entre decisiones tecnológicas y decisiones morales, entre diseño técnico y concepción moral que trata de recoger el concepto de público recursivo:

> This article goes into some technical detail on one particular corner of technical infrastructure that we use regularly, and looks at ways that architectural choices shape the social forces and structures attached to the infrastructure. But this is just one small corner. Most technological protocols we adhere to have social ramifications which are worthy of consideration. [...]
>
> If we make these social decisions in solidarity with each other, together we can build towards an egalitarian, democratic, non-hierarchical culture that spans the globe. The alternative is a fragmented society, where we are connected only to each other by the mechanisms of financial and cultural control, subjected to the whims of a small, powerful elite. So let's get to work! (Kahn Gillmor, 2007, pág. 68).

[34] Es interesante al respecto Hess (2006), donde se discuten diferentes maneras técnicas de controlar accesos y permisos, y su aplicación a diferentes partes de la infraestructura de Debian.

Esta doble dimensión moral y técnica es también, como veremos, central para los conceptos de «procomún» (véase la sección 5.2.4, *Procomún y cultura libre*), y «colectivo» (véase la sección 6.1, *De la comunidad al colectivo sociotécnico*).

3.3.2.1 Desarrollo en abierto: la apertura hacia el exterior.

Uno de los sentidos más importantes de la apertura que estamos tratando es la apertura del código, como recoge la expresión «código abierto». Pero quisiera centrarme ahora en otro sentido de la apertura, fundamental también para entender la naturaleza del Proyecto Debian como público recursivo. Me refiero a la apertura, como vimos en la introducción, de las prácticas sociales, de los procesos de decisión y del desarrollo de la distribución. Incluso de los problemas de la distribución, como queda recogido en el punto 3 del *Contrato Social* (apéndice A, *Contrato social*), citado continuamente por los miembros de Debian:

> **We will not hide problems**
>
> We will keep our entire bug report database open for public view at all times. Reports that people file online will promptly become visible to others.

Como es ya un lugar común en torno al *software* libre, esto permite que se solucionen los fallos más rápidamente, permitiendo ayudar a quien esté interesado, mejorando la calidad del *software*:

> **sez:** [preguntado por si el Contrato Social y las DFSG tienen algún impacto sobre la calidad del *software*] Well, the Social Contract says that we are not going to hide problemas from our users. It means that there is more light for the imperfect parts of the system, means that there are more opportunities for it to be fixed by those that care about it.

La no ocultación de los problemas se refiere explícitamente a los *bugs*, pero su sentido es mucho más extenso. El Proyecto Debian se caracteriza por su empeño

en mantener siempre abierta la posibilidad de abrir las *cajas negras* que, como vimos en la sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*, caracteriza el desarrollo técnico.

Para facilitar la tarea de unir las diferentes fuentes de datos disponibles sobre el Proyecto Debian, en el año 2008 comenzó el desarrollo de la *Ultimate Debian Database*, una base de datos centralizada donde se importan los datos situados en diferentes fuentes de la infraestructura del Proyecto, permitiendo a los usuarios de dicha base de datos acceder fácilmente a tales datos y combinarlos:[35]

> The Ultimate Debian Database (UDD) is a data warehousing solution for data sources about the Debian project. It was born to solve the so called «data hell» problem— an intrinsic difficulty in correlating distribution data to highlight package and maintainer metrics which are relevant for Quality Assurance (QA)—, but it is by no means QA specific. UDD is as suited for data mining research as it is for QA, which is in fact purpose-specific data mining. The generality of UDD design choices, together with the research attention on Debian which is not fading after 15 years of history, offers an unparalleled easy to use platform to establish facts about Debian. From an ethical point of view, UDD makes Debian be a better citizen in both the FLOSS and research ecosystems, by providing an open interface to all of its data—a feature that company-based distributions are usually unwilling to offer. That aspect can be leveraged by derivative distributions to import Debian data in their own infrastructures (Nussbaum y Zacchiroli, 2010).

Lo destacable para la discusión actual es que esta base de datos tiene por objetivo principal facilitar el trabajo de mantenimiento interno de la distribución, pero también pretende facilitar el acceso a los datos relativos a la distribución a los investigadores y a cualquier interesado en conocer su desarrollo. Como dice uno de los autores en una entrada de su *blog* (Zacchiroli, 2010e):

[35] Véase Udd.debian.org (s.f.). Para una explicación más completa de su historia y de la motivación para construir dicha base de datos, véase Nussbaum y Zacchiroli (2010), escrito por dos Desarrolladores de Debian.

> the main target of the paper is the community of scientists doing **data mining on software repositories**. For them, UDD offers a valuable entry point to Debian «facts», as data sources reflected in the database are easily joinable together and to some extent already validated by other UDD users (e.g. QA people) [...]
>
> I've already noted in the past how that is also related to the **culture of freedom** that in Debian we value not only in our software, but also in our infrastructure and procedures.

Curiosamente, de manera irónica, en la sección de agradecimientos aparece entre otros el agradecimiento para la elaboración del artículo a «the "German cabal" and Debian System Administrators for their UDD hosting and support». Esto provoca el comentario al *post* por parte de otro Desarrollador: «just remember, TINC», a lo que responde el autor «Sure! ... and in particular there is no Debian cabal which is taking over the scientific research community».

Las ironías sobre la existencia de cábalas son, como hemos visto, una forma de expresar el rechazo a la falta de transparencia. En Debian, la prevención ante la concentración de poder en pequeños grupos lleva a buscar la mayor transparencia posible. De forma curiosa, esto se aplica sobre todo a los encuentros en persona, mucho más que a la comunicación electrónica. En una comunidad que se comunica fundamentalmente por medios electrónicos, salirse de la comunicación *online* limita la apertura y la transparencia. Esto es así fundamentalmente porque una característica de los medios de comunicación y colaboración del Proyecto Debian es que son abiertos, accesibles para cualquiera y quedan archivados para una posterior consulta.[36]

En 2010, el entonces Líder del Proyecto proponía en su *blog* unas normas para evitar precisamente que las reuniones presenciales, cara a cara, levantasen dudas sobre la transparencia, más aún cuando se invierte dinero del Proyecto en ellas. El *post* se llama precisamente «How to have a (Debian) summit without turning into a secret cabal» (Zacchiroli, 2010f):

[36] Véase la sección 1.3, *Cuestiones de metodología*.

> It seems rather uncontroversial that sponsoring various kinds of Debian **meetings** (conferences, sprints, BSPs, etc.) is a good way to spend, actually **invest, Debian money**.
>
> Historically, that has not always saved the Debian community from muttering about «cabal-ish» meetings in very few specific occasions. (No, there is no cabal, in case you wonder.) I've always believed in the good faith of people and I don't think that we have ever had «secret meetings» on purpose. Nevertheless the question of **how to have meetings in a community-compatible way** is a sound one. [...]
>
> In particular, organizers have to carefully balance the high efficiency that meetings offer (e.g.: communication bandwidth is higher than when working remotely, people have less distractions, more enthusiasm, more fun!, etc.) with the **risk of cutting out the rest of the community** which cannot attend the meeting, for whatever reason.

Básicamente, lo que propone a continuación es mejorar la comunicación: anunciar las reuniones y ofrecer resúmenes de lo discutido y decidido. El lugar preferido es la lista de correo `debian-devel-announce`, dedicada a anunciar cuestiones que tienen que ver con el desarrollo de la distribución, y a la que sólo pueden escribir los Desarrolladores de Debian (Debian-devel-announce, s.f.). En esta lista son frecuentes los mensajes con el tema «Bits from the ...» diferentes equipos o grupos.

Hay que recordar en este punto uno de los componentes centrales de la infraestructura de Debian, el *Bug Tracking System* (véase el cuadro 2.3, *Bug Tracking System*). Como vimos, es también una infraestructura técnica completamente abierta, a la que cualquiera puede acceder y utilizar. También son fundamentales para una consideración de la infraestructura técnica y su papel en la constitución de un público recursivo los repositorios en los que se desarrolla el código que forma parte de la distribución. Ambos son centrales porque son los puntos principales donde se desarrolla la actividad fundamental del Proyecto Debian: el desarrollo y mantenimiento de los paquetes de *software* que forman la distribución. Sin embargo, es más adecuado examinar el funcionamiento de los repositorios en

la sección 4.8, *Repositorios de* software *libre*, para poder dar cuenta de toda su importancia.

3.4 Doocracia y Público Recursivo.

Podemos ver ahora que doocracia y público recursivo son dos caras de la misma moneda. Dos conceptos que nos sirven para describir y entender los mismos procesos sociales. Debido precisamente a que Debian funciona como una doocracia, en algunas ocasiones existe también la posibilidad de *hacer* una infraestructura o servicios alternativos:

> **micah:** When I was working on this security problem with the Security Team, there were some people who were setting up some alternative infrastructure, like other build daemons and volatile, backports, archive and this sort of things. They were being done by other people who are not DSA, who are not FTP Masters, who are not Release Managers, because those people in those positions were not interested in doing this work. And these people, they went and just set up their own machines and started doing [...] And these people who are setting up this alternative infrastructure were saying «we are building this so if somebody decides to cut everybody off, because they are pissed off or whatever, now we have something that we can work with».

Recordemos que en la caracterización de Kelty, lo que distinguía a los públicos recursivos de otras formas de organización era «their focus on the radical technological modifiability of their own terms of existence» (Kelty, 2008, pág. 3). A esta modificabilidad tecnológica es a lo que alude también el concepto de doocracia. Ésta expresa la capacidad de crear a partir de la propia acción, y de los cambios que ésta produce, una actualidad normativa. Expresa la creación de capacidad de decisión a partir de la asociación con dispositivos tecnológicos, que reconfiguran el marco de acción, y con él el reparto y las posibilidades de agencia. Tanto la doocracia como la formación de un público recursivo comparten el ser formas de constitución de nuevos actores, en los que las entidades que se asocian traducen mutuamente sus fines. El público recursivo surge mediante su estrecha

asociación con la infraestructura que lo mantiene, que hace surgir a ese público como un actor nuevo, con fines diferentes.

La doocracia necesita pues de medios abiertos en los que encontrar su lugar. Sólo en tanto sean abiertos podrá una acción modificarlos y afirmarse en su papel configurador de normas y convenciones. Dar lugar a una forma de organización basada en la doocracia necesita que se constituya también como público recursivo. En nuestro caso, la capacidad de un miembro de Debian para decidir está en relación con la modificabilidad de la infraestructura técnica y los dispositivos tecnológicos en los que se produce su acción. Es decir, con su apertura. Apertura en este contexto no se refiere exclusivamente a que algo sea público, sino también a que sea modificable. A que se puedan abrir las «cajas negras» que constituyen la infraestructura, no sólo para examinarlas sino también para modificarlas. Por definición, en el caso del *software* libre, la apertura nos remite siempre a la posibilidad de abrir las cajas negras de los dispositivos tecnológicos que produce, para analizar su funcionamiento y modificarlo.

Al mismo tiempo, para que se pueda constituir un público recursivo es necesario también que exista alguna forma de doocracia. Por la misma razón, porque sólo en tanto todas las capas que lo componen sean efectivamente modificables a iniciativa de los individuos que están constituyendo ese público recursivo podrá entenderse como un espacio autoproducido, autofundamentado. Aunque no usa el concepto de doocracia, esta cuestión aparece también en la consideración de Internet como público recursivo en Kelty (2005a).

Para acabar esta sección, quisiera insistir en la idea de que, según hemos visto, la noción de público recursivo apela tanto a la idea de un orden técnico como de un orden moral o social. En la sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*, vimos cómo la distinción entre un ámbito técnico y otro social era una distinción producida a partir de la propia acción social de los miembros de Debian. Podemos ahora afirmar que se produce esa distinción en buena parte porque la acción social se da en el marco de un público recursivo, en el que la acción social transforma simultáneamente los dispositivos técnicos en los que se expresa y las normas convencionales que la constituyen. En la constitución y configuración del orden normativo intervienen también los dispositivos tecno-

lógicos, así como la concreta configuración que éstos adoptan depende también de elecciones normativas. Como afirmaba Kahn Gillmor, «Technical Architecture shapes Social Structure». También a la inversa. O mejor, sólo tras la acción social podemos establecer esa diferencia. Veamos en el capítulo siguiente ese doble carácter del *software* como dispositivo tecnológico.

4 El carácter híbrido del *software*. *Bugs*, *hackers*, paquetes y repositorios.

> *But in the real world of ethnographic detail and anthropology from a pragmatic point of view, life and code are much more full of intrigue, puzzling, and gaming, involving plenitudes of passions and reasons, hacks and bugs, patches and work-arounds, values and interests, social imaginaries and institutional demands. It is a world, in Deleuze and Guattari's (1980) vocabulary, of assemblages rather than unified machines.*
>
> —Fischer (2007, pág. 571).

Es el momento de detenernos a examinar de manera más general algunas características de los objetos que dan consistencia al Proyecto Debian como entramado sociotécnico y como colectivo (capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*): el *software* o código. Si el dualismo entre lo técnico y lo social se producía en la práctica social (capítulo 2, *Debian como trama sociotécnica*), aquí consideramos el híbrido sociotécnico a partir del que se produce. Será además el objeto que hace posible la doocracia al establecer relaciones con él (capítulo 3, *Formas de gobierno. Doocracia, agencia y apertura*) y el objeto del don (capítulo 5, *El don del* software *libre*).

En realidad, los objetos y dispositivos tecnológicos que juegan un papel en la constitución del colectivo son innumerables: servidores, repositorios, *emails*, canales de IRC, informes de fallos, *scripts*, o lenguajes de programación, por nombrar sólo una pequeña parte. La estrategia que se desarrolla en este trabajo es introducirlos cuando es necesario en la descripción de la acción social analizada. Pero puede ser muy ilustrativo dedicar un capítulo al objeto alrededor del cual se articulan y toman consistencia las prácticas sociales en Debian, el código. Y es que Debian es básicamente una distribución de *software* libre, lo que hace en primer lugar es distribuir *software*, código.

Una breve precisión terminológica. Así distingue el *Linux Information Project* (Linfo.org, s.f.[a]) los términos *software* y código fuente:

> *Software* is a generic term for *programs* that are used by computers and other products that contain logic circuitry (i.e., embedded systems). In a broader sense it can also refer to all information (i.e., both programs and data) in electronic form, and it can provide a distinction from *hardware*, which refers to media and systems on which software can exist and be used (Linfo.org, s.f.[b]).
>
> *Source code* (also referred to as *source* or *code*) is the version of *software* as it is originally written (i.e., typed into a computer) by a human in *plain text* (i.e., human readable alphanumeric characters) (Linfo.org, s.f.[c]).

El término *software* es más amplio que el de *código* fuente o simplemente código, puesto que lo incluye. El código fuente es la versión del *software* legible por un ser humano, mientras que *software* se aplica también a su versión binaria, legible sólo por un ordenador o procesador. En tanto ambos refieren al conjunto de instrucciones que ejecuta un ordenador, en muchas ocasiones son intercambiables y hablaremos indistintamente de código o *software*. Como vimos en la introducción, la definición de *software* libre requiere la libertad de acceso, modificación y distribución no sólo de la forma compilada, binaria, del *software*, sino también la del código fuente que lo origina.

Por otra parte, para analizar las diferentes dimensiones culturales del *software* y su papel en la constitución de redes sociotécnicas es necesario tener en cuenta la especificidad del código fuente. Porque al ser lo que produce un programador, y al ser legible directamente por seres humanos, revela una mayor riqueza de conexiones y asociaciones. Fischer lo expresa del siguiente modo en la cita que abre este capítulo, y que se aplica más propiamente al código fuente tal como se ha definido:

> But in the real world of ethnographic detail and anthropology from a pragmatic point of view, life and code are much more full of intrigue, puzzling, and gaming, involving plenitudes of passions and reasons, hacks and bugs, patches and work-arounds, values and interests, social imaginaries and institutional demands. It is a world, in Deleuze and Guattari's (1980) vocabulary, of assemblages rather than unified machines [...] (Fischer, 2007, pág. 571).

Es decir, el *software* es complejo y multidimensional, una entidad heterogénea en su constitución más que algo unificado. Generalizando lo que veremos especialmente en la sección 4.4, *El código como objeto híbrido*, la consideración del *software* como objeto híbrido nos permitirá entender cómo y por qué es parte constituyente del colectivo que describimos, al tiempo que posibilita la disolución de los dualismos que dificultan su comprensión: el *software* es un objeto híbrido porque no se adecua a los repartos dualistas entre las herramientas y los conocimientos, entre lo técnico y lo social, lo instrumental y lo expresivo, o el don y la mercancía, entre otros. También ayuda a comprender como funciona la doocracia y los públicos recursivos.

En los últimos años ha aumentado el interés por el estudio del *software* y del código desde las ciencias sociales, las humanidades y los estudios culturales, y están apareciendo campos nuevos como los *software studies*, los *critical code studies*, los *new media studies* o los *game studies*. Aunque tienen una perspectiva diferente a la empleada en esta investigación, algunas de las referencias básicas de estos nuevos campos que se han tenido en cuenta son Fuller (2003, 2008b), Mackenzie (2006), Manovich (2001, 2013) y Marino (2006). Para una consideración

más general sobre la etnografía de la infraestructura, véase Star (1999). Para una consideración general de las implicaciones sociopolíticas de los artefactos técnicos, véase Winner (1986).

4.1 La Red como metáfora. Unix.

La naturaleza específica del *software* y del código hace que de alguna manera algunos conceptos de la Teoría del Actor-Red de Callon y Latour se presenten aparentemente como los más adecuados para dar cuenta de sus características y describir las prácticas asociadas a su producción y distribución. Esto se debe al menos en parte al hecho de que estos conceptos, que en relación a tecnologías anteriores aparecen como metáforas y su aplicación supone un cierto esfuerzo de la imaginación, en el caso del *software* y del código se aplican a veces casi literalmente. Aparece como algo evidente que un *programa* de *software* es un *programa* de acción. O que una herramienta para *escribir* código es un dispositivo de *inscripción*, por ejemplo. Recordemos una cita de Latour que vimos en relación a la disputa sobre el *firmware* no libre:

> The program of action is the set of written instructions that can be substituted by the analyst to any artifact. Now that computers exist, we are able to conceive of a text (a programming language) that is at once words and actions. How to do things with words and then turn words into things is now clear to any programmer (Latour, 1992, pág. 255).

La cuestión es que el programa sigue siendo en sí mismo un conjunto de instrucciones escritas, no sólo su concreción compilada, binaria, o su descripción. El código literalmente hace cosas con palabras. *Es ya* un conjunto de enunciados:

> How can the prescriptions encoded in the mechanism be brought out in words? By replacing them by strings of sentences (often in the imperative) that are uttered (silently and continuously) by the mechanisms for the benefit of those who are mechanized: do this, do that, behave this way, don't go that way, you may do so, be allowed to go there. Such

> sentences look very much like a programming language (Latour, 1992, pág. 232).

Sin embargo, esos enunciados que interesan al investigador no se reducen a los enunciados que aparecen explícitamente en el código informático, por lo que sigue siendo necesario reconstruir todo su entramado y vinculación con otros conjuntos de enunciados. El *software* es un objeto híbrido, complejo, porque la escritura de código supone enlazar en una unidad, el *software* producido, elementos muy heterogéneos: compiladores, intérpretes, bibliotecas externas, ficheros externos, metodologías de desarrollo, herramientas de depuración, estilos y lenguajes de programación, sistemas de desarrollo, de preparación de paquetes, tests, y sistemas de control de versiones, por mencionar algunos.

Una de las metáforas dominantes aparece en la misma denominación de la teoría: la red. Esta teoría aparece así como natural e inmediatamente aplicable a las redes informáticas de comunicación. La tentación es pensar que el modelo o paradigma de las redes en el sentido de la Teoría del Actor-Red es Internet. Pero el concepto de red es un constructo metodológico y epistemológico, que no presupone que su objeto se configure material y topológicamente como una red de comunicación entre diferentes nodos. Como dice Latour (2005, págs. 128-129), lo que tiene forma de red es la descripción, no lo descrito. Por supuesto, como cualquier otro artefacto sociotécnico, es muy interesante una descripción de Internet u otras redes de comunicación digital desde esta perspectiva. Pero para entender cómo se integran en las redes que nos interesan, en la descripción. No hay que dar por supuesto que la descripción reflejará una estructura de asociaciones en la que los elementos enlazados son máquinas o usuarios discretos, sino que habrá que determinar empíricamente y *a posteriori* cuáles son los agentes (un ejemplo muy simple: un servidor es una máquina conectada a otras, pero en un servidor físico pueden funcionar diferentes tipos de servidores, procesos autónomos que se pueden comunicar entre sí y con otros, haciendo irrelevante que estén o no en la misma máquina). El hecho de que se comuniquen a través de una red como Internet no hace que formen más parte de una red (en sentido etnográfico) que la asociación entre diferentes elementos locales que ocupan un mismo espacio.

Un paradigma del concepto de red más adecuado, un mejor objeto concreto para pensarlo o ilustrarlo, bien podría ser el sistema operativo (o familia de sistemas) UNIX (el sistema sobre el que se basan los sistemas GNU/Linux, como Debian). Una de las razones por las que UNIX es una buena metáfora es el interés que ha existido no sólo por su calidad tecnológica, sino también por su historia (que se remonta a principios de los 70), su filosofía e incluso su cultura. Todos estos aspectos están íntimamente relacionados con su diseño técnico, no son añadidos a un contenido tecnológico neutro:

> The UNIX operating system is not just a technical achievement; it is the creation of a set of norms for sharing source code in an unusual environment: quasi-commercial, quasi-academic, networked, and planetwide.
>
> [...]
>
> Sharing source code in Free Software looks the way it does today because of UNIX. But Unix looks the way it does [...] because *sharing produces its own kind of order*: operating systems and social systems. [...] UNIX is not just an operating system but a way of organizing the complex relations of life and work through technical means; a way of charting and breaching the boundaries between the academic, the aesthetic, and the commercial; a way of implementing ideas of a moral and technical order (Kelty, 2008, págs. 141-142).

Una figura muy importante en la formulación de, y la reflexión sobre, la cultura y la filosofía de UNIX ha sido Eric Raymond, un influyente programador y activista del *software Open Source* (véase la sección 5.2.2, *Sobre el sentido de «libre» en el* software *libre y la cultura libre*). En su libro *The Art of Unix Programming* señala la importancia de la tradición cultural de UNIX, y su estrecha relación con la cultura *hacker* y el *software* libre:[1]

[1] En realidad Raymond aquí habla del movimiento *Open Source*, que hace surgir en 1998, año de fundación de la *Open Source Initiative* (entre otros por él mismo), olvidando todo el movimiento del *software* libre inspirado por Richard Stallman (véase la sección 5.2.2, *Sobre el sentido de «libre» en el* software *libre y la cultura libre*). Sí parece hacer referencia al surgimiento de Linux a principios de los 90.

> The Unix tradition is an implicit culture that has always carried with it more than just a bag of technical tricks. It transmits a set of values about beauty and good design; it has legends and folk heroes. Intertwined with the history of the Unix tradition is another implicit culture that is more difficult to label neatly. It has its own values and legends and folk heroes, partly overlapping with those of the Unix tradition and partly derived from other sources. It has most often been called the "hacker culture", and since 1998 has largely coincided with what the computer trade press calls "the open source movement". The relationships between the Unix tradition, the hacker culture, and the open-source movement are subtle and complex. They are not simplified by the fact that all three implicit cultures have frequently been expressed in the behaviors of the same human beings. But since 1990 the story of Unix is largely the story of how the open-source hackers changed the rules and seized the initiative from the old-line proprietary Unix vendors. Therefore, the other half of the history behind today's Unix is the history of the hackers (Raymond, 2003, pág. 66).

La «filosofía de UNIX» ha sido elaborada, desarrollada y descrita por muchos autores, desde los creadores del sistema operativo Ken Thompson y Dennis Ritchie. Hay muchas maneras de caracterizarla y sintetizarla, pero una de las más importantes es la propuesta por Kernighan y Pike, dos importantes programadores y contribuyentes al desarrollo de UNIX, muchas veces repetida:

> Although that philosophy can't be written down in a single sentence, at its heart is the idea that the power of a system comes more from the relationships among programs than from the programs themselves. Many UNIX programs do quite trivial things in isolation, but, combined with other programs, become general and useful tools (Kernighan y Pike, 1984, pág. viii).

La colaboración entre pequeños programas que hacen sólo una cosa pero la

hacen bien es quizás el principio de diseño más importante de UNIX.[2] Cuando se necesita una funcionalidad más compleja se prefiere combinar adecuadamente pequeños programas que desarrollar uno más complejo. Algunos aspectos de UNIX que permiten esto son el uso de tuberías (*pipes*, elementos que conectan dos programas haciendo que la salida de uno sea la entrada de otro) y redirecciones; el hecho de que todo en UNIX sea un fichero, legible y escribible por los programas (y a menudo por los humanos), facilitando el intercambio de información; o el desarrollo e implementación de estándares y protocolos que permitan el intercambio y la comunicación, por poner algún ejemplo. Todo esto hace que en el diseño del sistema operativo sea esencial la capacidad de establecer relaciones nuevas y creativas entre sus componentes. Los diferentes elementos adquieren su potencia sólo por medio de la red que los constituye. Encontramos aquí una ilustración de los conceptos de delegación y reclutamiento de no humanos, y del modo en que así se modifican, se traducen los fines de los agentes implicados. Esta combinación de herramientas se produce fundamentalmente en la línea de comandos. De ahí la preferencia por éstas entre los miembros de Debian, en tanto les da más agencia residencial (véase la sección 2.2.8, *Agencia y semiótica*) que una interfaz gráfica.

La actividad de programar en sí misma, y aún más la de programar un sistema operativo, es una práctica de construcción de redes y colectivos, porque supone la puesta en marcha de muchos actantes (véase la Sección 4.2, *El código como actante*) a partir del reclutamiento, la reunión y articulación de otros con orígenes heterogéneos (bibliotecas de *software*, otros programas, agentes humanos, ...). Esto es aplicable tanto al trabajo de programación en un determinado entorno como a la creación de ese mismo entorno, por ejemplo UNIX o GNU/Linux. Así, un elemento fundamental de la historia de UNIX es su estrecha relación con C, uno de los lenguajes de programación más usados e influyentes. C fue creado por uno de los creadores de UNIX, Dennis Ritchie, precisamente para reescribir UNIX. Este paso de la programación de UNIX desde el lenguaje ensamblador (lenguaje de bajo nivel, dependiente de la arquitectura física del procesador, y por lo tanto difícilmente portable) al lenguaje de alto nivel C tuvo como consecuencia que el código de UNIX pudiese ser portado a otros sistemas físicos, a otros procesadores

[2] La entrada de Wikipedia sobre *Unix philosophy* es un buen punto de entrada sobre estas cuestiones, en Wikipedia (s.f.[e]).

diferentes.

Este desarrollo histórico es fundamental, ya que prosigue el proceso de separación entre *hardware* y *software*, posibilitando el uso de los mismos protocolos y código en diferentes plataformas de *hardware*. Y es que el *software* como tal no ha existido desde el inicio del desarrollo de los ordenadores, como algo separado del soporte físico en el que funciona.[3] Esta separación es indisociable de la separación entre código fuente y código binario, ligado a la arquitectura concreta de la máquina. El desarrollo de C y de UNIX son un paso importante en esta historia. El lema de Debian, «El Sistema Operativo Universal» (que se refiere entre otras cosas a la aspiración a funcionar en todas las arquitecturas posibles; de hecho, es el sistema operativo que funciona en un mayor número de arquitecturas diferentes), refleja la aspiración a llevar este proceso al extremo. Se ha extendido la idea del ordenador como máquina universal (que se remonta a Turing) porque puede realizar cualquier tarea computable, a la idea del *software* universal como proceso que se puede realizar en cualquier ordenador o arquitectura.

Manovich (2013) destaca la importancia de este proceso de separación para la aparición y desarrollo de los «nuevos medios», por la flexibilidad que posibilita:

> In short, *"new media" is "new" because new properties (i.e. new software techniques) can always be easily added to it*. Put differently, in industrial (i.e. mass-produced) media technologies, "hardware" and "software" were one and the same thing. [...] What differentiates a modern digital computer from any other machine –including industrial media machines for capturing and playing media– is separation of hardware and software (Manovich, 2013, pág. 92).

Volvamos a las características de UNIX. Al estar escrito en un lenguaje de alto nivel, el código de UNIX puede ser más fácilmente entendido, y por lo tanto es más fácil colaborar en su desarrollo. Todos estos rasgos facilitaron la formación de una comunidad que compartía sus ideas y desarrollos. Dos revisiones de

[3] El primer uso del término *software* es tan tardío como 1958. Para algunos elementos de la historia del *software* y del código fuente, véanse Fuller (2008a) y Krysa y Sedek (2008).

esta historia muy ilustrativas se encuentran en el capítulo *Sharing Source Code* de Kelty (2008) y en *The Early History of Open Source* de Weber (2004).

Una de las principales características de UNIX como sistema operativo es entonces su flexibilidad, que lo hace especialmente adecuado para los *hackers*: «Given that it is considered a flexible partner, Unix is loved by hackers» (Coleman, 2013, pág. 36). El nombre de la sección donde aparece esta cita, tomado de Stephenson, refleja perfectamente la significación cultural de un sistema como UNIX: «UNIX as "Our Gilgamesh Epic"».

Pero UNIX y su diseño no son sólo apreciados, sino también usados para pensar su propia constitución como grupo. En la siguiente cita, un Desarrollador de Debian responde a una pregunta sobre una propuesta (que será analizada en la sección 6.3, *Segunda controversia: ¿Quién es miembro de Debian?*) para introducir en el Proyecto diferentes categorías de miembros, estableciendo una analogía explícita entre el funcionamiento del Proyecto y el de una máquina UNIX:

> **mones:** Además va totalmente en contra de la filosofía de los entornos UNIX, que es siempre un entorno colaborativo. El tema de los permisos, de los grupos, se inventó para que la gente pueda colaborar en las mismas áreas. Lo veo en la misma línea filosófica. Hasta ahora tener *Debian Developers* significa que sólo tienes un grupo en la máquina. Ninguna máquina tiene sólo un grupo de usuarios. Y realmente ya existe, porque no todos los *Developers* tienen acceso a todos los recursos del proyecto.

Este último rasgo de UNIX se refiere a su implementación de un sistema de usuarios y grupos con diferentes privilegios y permisos de acceso en una máquina. De hecho, como vimos en el Capítulo 3, *Formas de gobierno. Doocracia, agencia y apertura*, y seguiremos viendo en la sección 6.3, *Segunda controversia: ¿Quién es miembro de Debian?*, en buena parte los derechos y privilegios de acceso de los miembros del Proyecto Debian se definen en gran medida por los permisos de usuario que tenga y los grupos a los que pertenezca en la infraestructura técnica

del Proyecto.[4]

Por todo esto, UNIX es un buen paradigma para la Teoría del Actor-Red. Sobre todo, porque hay una clara articulación (y afinidad) entre su diseño técnico y las formas sociales con las que se ha vinculado. Los principios de modularidad, apertura, transparencia o flexibilidad son tanto principios de organización técnica como de organización social. O posiblemente deberíamos decir, a partir de lo expuesto en la sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*, que distinguir ambos elementos sólo es posible después de un trabajo de separación. Muchos de sus rasgos han sido heredados por los sistemas GNU/Linux. Su diseño, historia y cultura han influido en el desarrollo histórico de la cultura *hacker*, del movimiento del *software* libre y por lo tanto de Debian, y son invocados como argumentos en numerosas discusiones.

Nos detendremos a continuación en dos conceptos desarrollados por Latour que me parecen especialmente útiles para comprender el papel del *software* y del código: *actante* y objeto *híbrido*. Entre las secciones dedicadas a ellos, se introducirá una discusión sobre las dimensiones instrumental y expresiva del código.

4.2 El código como actante.

Al hablar del *software* o del código como actante lo que se pone de relieve es su capacidad de agencia, su papel en la constitución y configuración de las cadenas de acción que se dan en los procesos y prácticas sociales estudiados. Ya vimos en la sección 2.1.6, *Agentes, humanos y no humanos*, cómo hay que recurrir a diferentes tipos de agentes para describir los procesos sociales. Si esta atribución de agencia es una antropomorfización, ésta se da ya en el lenguaje propio de la informática (no sólo, pero especialmente en sistemas de tipo UNIX), donde son de uso frecuente términos como cliente, servidor, demonio o agente para referirse a dispositivos tecnológicos o procesos con los que se interactúa, o que interactúan entre ellos.

Por ejemplo, en el texto de Kahn Gillmor comentado al final del capítulo anterior podemos encontrar un caso claro en que los actores con los que interac-

[4] Para una visión crítica de esta situación, véase Hess (2006).

tuamos no son siempre humanos:

> [...], there could still be problems: some nasty group could be intercepting your communications, and claiming to be the group you actually want to talk to. This isn't veering into paranoia here: the global network is very flexible; it relies on wide-scale co-operation; and the malicious actors are often tireless and conscienceless machines, not individual humans (Kahn Gillmor, 2007, pág. 57).

O veamos la siguiente cita de Donald MacKenzie, también referida al contexto de la seguridad informática:

> In document-based systems, human users were implicitly trusted not to "write down" in this way [a un documento con una clasificación de seguridad más baja], but the broadening of the notion of "subject" to include computer processes and programs raised a new issue: the risk that a hostile agent might insinuate a "Trojan horse" into a trusted system (MacKenzie, 2004, pág. 164).

MacKenzie está discutiendo un modelo específico de seguridad informática. Lo importante de esta cita es que nos muestra que, efectivamente, en el campo de la informática el concepto de agente, actor o incluso sujeto se amplía necesariamente para incluir programas y procesos. Sobre todo en el campo de la seguridad informática, incluso a nivel del sistema operativo, las nociones de «sujeto» y «agente» han de ser redefinidas y ampliadas. Esto no es sólo una forma de hablar. En buena medida, implementar la política de «no write down» *requiere* considerar los programas y procesos como agentes, y como relativamente autónomos. En el modelo de seguridad Bell-LaPadula (MacKenzie, 2004, cap. 5), la distinción básica era la que se establecía entre sujetos (agentes que tienen acceso a los objetos; tienen un grado de autorización) y objetos (que tienen una clasificación de seguridad), y la preocupación fundamental consistía en asegurar que ningún sujeto podía acceder a un objeto con una clasificación de seguridad más alta, ni escribir

un documento con una clasificación más baja. Pero los procesos que pueden funcionar en un ordenador (sobre todo en un ordenador conectado a una red) no se ajustan a ese simple esquema, porque no son simples objetos:

> Applying the Bell-LaPadula model to a network is not straightforward: it is, for example, not always clear which entities should be considered "subjects" and which "objects" (MacKenzie, 2004, pág. 190).

Realizan acciones, y en cuanto a la seguridad al menos han de ser tratados como sujetos capaces de hacer cosas con la información. Esto se puede generalizar a las arquitecturas servidor-cliente o al uso de demonios. Responden autónomamente, nos hacen hacer cosas distintas. Crean conexiones y asociaciones nuevas en el entorno del computador.

A continuación veremos dos conceptos que permiten dar concreción empírica a esta atribución de agencia, y su relación con dos elementos de distinto orden pero importantes para entender nuestro campo, las figuras del *bug* y del *hacker*.

4.2.1 Agenciamientos.

No se trata de afirmar que los objetos técnicos tengan por sí mismos y de manera aislada agencia, ni mucho menos intencionalidad. Pero tampoco se puede reducir la agencia a una propiedad interna de los seres humanos. Según Callon,

> Agency as a capacity to act and to give meaning to action can neither be contained in a human being nor localized in the institutions, norms, values, and discursive or symbolic systems assumed to produce effects on individuals. Action, including its reflexive dimension that produces meaning, takes place in hybrid collectives comprising human beings as well as material and technical devices, texts, etc.
>
> [...]
>
> Action is a collective property that naturally overflows. To be attributed to a particular agency, it has to be framed (Callon, 2005, pág. 4).

La acción propiamente entonces es una propiedad colectiva, que surge de la asociación entre elementos heterogéneos, humanos y no humanos. En los casos que estamos considerando, una forma importante de establecer estas asociaciones es precisamente la doocracia, que como hemos visto depende en muchas ocasiones de la asociación con una entidad técnica, creando así un nuevo actor. Atribuir agencia a un dispositivo informático, por ejemplo, puede ser una manera conveniente de describir, enmarcándolas y simplificándolas, determinadas acciones, y esta atribución no deben ser tomada literalmente en el sentido de que esos dispositivos tengan agencia por sí solos, considerados aisladamente. Pero lo mismo se puede decir de la atribución exclusiva a individuos humanos. La agencia reside en *actores-redes*, o más precisamente, en *agenciamientos*:

> I use the french word *agencement*, instead of arrangement, to stress the fact that agencies and arrangements are not separate. *Agencements* designate socio-technical arrangements when they are considered from the point view of their capacity to act and to give meaning to action (Callon, 2005, pág. 4).

Como MacKenzie (2009b, p. 20 ss.) señala al discutir esta concepción, en la idea de agenciamiento se combinan la noción de ensamblaje o configuración, y la de agencia. Otra virtud de este concepto es que reúne, en la definición de Callon, las capacidades de actuar y de crear significado, como vimos en la concepción semiótica de la agencia que apareció en la sección 2.2.8, *Agencia y semiótica*.

Pero el principal beneficio del concepto de agenciamiento es que nos obliga a tener en cuenta la infraestructura tecnológica que es condición de posibilidad de la acción y de la configuración de los actores. Hardie y MacKenzie lo señalan a propósito de los actores económicos, pero lo mismo podría decirse de los participantes en la creación y desarrollo de *software*:

> At the most basic level, the notion of *agencement* helpfully directs us to the conditions of possibility of economic actors: the often-ignored infrastructure that enables them to be the actors they are (MacKenzie y Hardie, 2009).

Lo que estamos describiendo en este trabajo de investigación son entonces agenciamientos: el BTS, un paquete de *software*, el archivo de Debian, el *New Maintainer Process*, las licencias, etc., son agenciamientos. Sólo se pueden describir los procesos y las prácticas sociales al hilo de la reconstrucción de la vinculación entre todas las entidades, de cualquier tipo, que influyen en ellos.

4.2.2 Mediadores e intermediarios. Las figuras del *bug* y del *hacker*.

Pero si el *software* es un actante o un actor debido a su inclusión en agenciamientos, lo es porque se comporta como un *mediador* y no como un *intermediario*. Así los distingue Latour:[5]

> Pero entonces dejan de ser simples intermediarios más o menos fieles. Se convierten en mediadores, o sea, actores dotados de la capacidad de traducir lo que transportan, de redefinirlo, de redesplegarlo, y también de traicionarlo. Los siervos han vuelto a ser ciudadanos libres (Latour, 2007, pág. 121).

Un mediador crea una diferencia inesperada, una transformación, una traducción. Su papel no es reducible a las intenciones de quién lo pone en movimiento. Excede siempre la intencionalidad, ofreciendo una resistencia, de los actores que se asocian con él, convirtiéndose por ello él mismo en un actante. Esta es una experiencia muy común en el uso de las tecnologías informáticas, como me decía un Desarrollador de Debian: «el ordenador no hace lo que quieres que haga, sino lo que le dices que haga». El problema, por supuesto, es que no siempre se consigue decirle precisamente lo que se quiere que haga, o se dice más o menos de la cuenta. O también:

> **bencer:** [...] con la informática no es posible, hay tantísimos niveles, que es imposible controlarlo todo. No puedes asegurar nunca que el

[5] Para una explicación detallada de la distinción entre *intermediario* y *mediador*, véase Latour (2007, págs. 118-123).

> *software* vaya a funcionar bien, debido a la complejidad. En la universidad te inculcan primero diseñar, luego escribir ... Pero hasta que no me pongo a escribir no sé a qué problemas me voy a enfrentar, es imposible que eso salga bien.

Según Callon (1984) la traducción es precisamente el proceso general de constitución de actores y colectivos (véase la sección 2.1.7, *Procesos de traducción*). Y en la Teoría del Actor-Red, la traducción es ante todo traducción de fines y objetivos. Es en este sentido en el que una asociación entre actores (constituyendo uno nuevo) es un proceso de traducción: cambian mutuamente sus fines. Por lo tanto esos procesos de traducción son esenciales para entender la distribución de la agencia: un agente no sólo tiene más agencia en tanto es capaz de movilizar más medios, sino también en tanto es capaz de modificar los fines de otros actores, haciéndolos pasar por sus propios fines.

Cuando esto ocurre, surge un nuevo agente a partir del enrolamiento de un agente por otro, a partir de su fusión en ese conjunto de acción. Cada uno de ellos puede tener su propio programa de acción (para este concepto, véase la página 121), pero se produce una traducción entre sus metas de tal modo que surge una meta compuesta:

> El mito de la herramienta neutral bajo completo control humano y el mito de un destino autónomo que ningún ser humano puede dominar son simétricos. [...] una tercera posibilidad: la de la creación de una meta nueva que no correspondería a ningún programa de acción de los agentes implicados. [...] llamé traducción a esta incertidumbre acerca de las metas. [...] He utilizado el término traducción con la intención de que signifique desplazamiento, deriva, invención o mediación: la creación de un lazo que no existía con anterioridad y que en cierta medida modifica a los dos iniciales (Latour, 2001, pág. 214).

Veamos a continuación dos de las figuras en las que se encarna este proceso general de traducción llevado a cabo por los mediadores, por los actantes.

La primera como ilustración, la segunda además por su papel sustantivo, mucho más importante. Dos figuras que, a raíz de esta imprevisibilidad del *software*, desmienten tanto el determinismo tecnológico como el social: la del *bug* (fallo o error; véase la sección 2.1, *Bug #573745*) y la del *hacker*. Los desmienten porque en el tipo de acción social que estamos considerando no se pueden dejar de lado, si se quiere entender algo, las complejas interacciones entre las acciones e intenciones de los individuos y las del *software*, más allá de las pretendidas por el programador. Esta complejidad e imprevisibilidad son el origen de una gran cantidad de *bugs* que no se pueden reducir a un error o un mal funcionamiento, sino que emergen de la interrelación entre el comportamiento esperable de diferentes componentes de un sistema complejo. Como dice Latour (2001, pág. 339) y sería evidente para cualquier Desarrollador de Debian, «Incluso un ingeniero informático que crea programas de soporte lógico se ve sorprendido por su creación tras haber escrito un par de miles de líneas de cadenas lógicas».

Esto no siempre es algo negativo, puesto que puede abrir nuevas posibilidades. De ahí la expresión recurrente «It's not a bug, it's a feature» que se suele repetir como broma o excusa. Pero realmente la línea de separación entre *bugs* y *features* es delgada, y que algo sea interpretado como un fallo o una característica deseable depende de cómo se relaciona con los fines del resto de agentes. En una charla del creador de Linux Linus Torvalds en la DebConf 14 (Meetings-archive.debian.net, s.f.[c]), señalaba, a propósito de la cuestión sobre si corregir o no un determinado *bug*, «if it is a bug that people relies on, it's not a bug, it's a feature». Lo que está describiendo a fin de cuentas es una situación en la que el supuesto fallo ha modificado de hecho las metas de alguien, al darle nuevas posibilidades no previstas. En este sentido, un *bug* puede ser un mediador en tanto que no es simplemente un defecto, una falta que resta de las intenciones del autor. Pero además muestra que también lo es el código en el que se produce, en tanto que muchos *bugs* no son un simple resultado de la falta de atención o de conocimientos, sino que son introducidos sólo por la relación compleja del código con diferentes sistemas con los que interactúa.

Pero si la lógica compleja del código hace inevitable la aparición de *bugs* que actúan en contra de la voluntad del programador (pero a veces, también, a favor de la voluntad de otros), también permite por la misma razón la creación de *hacks*

como actividad del *hacker*. Así que estas dos figuras no son independientes. En muchas ocasiones de hecho la posibilidad de un *hack* depende de la existencia de un *bug*.

La figura del *hacker* está muy relacionada con la del desarrollador de *software* libre, como hemos visto en la Sección 4.1, *La Red como metáfora. Unix*, en una cita de Raymond. Sobre la relación de los desarrolladores de *software* libre con la cultura *hacker*, véase especialmente Coleman (2013). Una discusión muy interesante, aunque centrada sobre todo en la relación con la criptografía, se halla en Wolf (2016), donde encontramos un desarrollo histórico del término *hacker* y su polisemia. El *hacking* no consiste sólo en usar el *software* de otro modo al que fue diseñado, sino también en crearlo y modificarlo. Una definición ya clásica de *hacker* es la que se encuentra en *The Jargon File* (Raymond y Steele, 2004). Entre las varias acepciones y otros datos, se encuentra esta primera definición:

> A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary. RFC1392, the Internet Users' Glossary, usefully amplifies this as: A person who delights in having an intimate understanding of the internal workings of a system, computers and computer networks in particular.

El *hacker* se define entonces principalmente por el interés por el conocimiento íntimo de las mediaciones tecnológicas que configuran un sistema. También por su utilización de formas nuevas e inesperadas. Ese uso es lo que podemos llamar un *hack*, como práctica específica del *hacker*. Un *hack* puede consistir en la inserción de un dispositivo lógico en una red de tal modo que cambia su naturaleza introduciendo nuevos elementos y conexiones (González de Requena, 2016). Un uso nuevo y creativo pasa por establecer una nueva asociación de entidades que en absoluto se limitan a artefactos técnicos: pueden incluir un algoritmo, una vulnerabilidad, un *bug* o característica no documentada, un vacío legal, alguna característica del sistema operativo o del *hardware*, un comportamiento no esperado de los usuarios, entre muchos otros. En definitiva, un agenciamiento compuesto por entidades heterogéneas.

La práctica social del *hacker* se puede entender entonces como un antiprograma (sobre los conceptos de programa y antiprograma, véanse las páginas 121 y 133), una redistribución de la distribución de competencias inscrita en un dispositivo tecnológico. Akrich lo explica en estos términos:

> technical objects define actors, the space in which they move, and ways in which they interact. Competences in the broadest sense of the term are distributed in the script of the technical object. Thus many of the choices made by designers can been seen as decisions about what should be delegated to a machine and what should be left to the initiative of human actors (Akrich, 1992, pág. 216).

Y redistribuye las competencias porque en su relación con la tecnología delega menos a los no-humanos que un usuario no experto, precisamente por su gusto por el conocimiento del funcionamiento de la misma. Observamos aquí casi una paradoja. Es cierto que, como afirma Latour (1992, pág. 232), cuanto menos conocimientos (*deskilled*) tiene un humano más experto (*skilled*) tiene que ser el no-humano en el que se delega parte de su acción. Pero el *software* complejo y potente le permite al *hacker*, usuario experto, mayor capacidad de elección y control de la acción. Más que una simple redistribución de competencias, lo que encontramos en las prácticas que estamos considerando es la necesidad de mantener abierta la posibilidad de tal redistribución. En el caso de Debian, esto se puede observar también en la tensión entre el intento de Debian de hacerle la vida más fácil al usuario, y el interés por ofrecer alternativas y permitir que cada usuario tenga una gran capacidad de elección.

En cualquier caso, no tomo aquí *hacker* como un término que haga referencia a un grupo o categoría bien definida de sujetos, sino como una posición subjetiva que une estrechamente una determinada relación con la tecnología (o mejor, como vimos al inicio del capítulo 2, *Debian como trama sociotécnica*, «un *hacker* es alguien que tiene una relación especial con la tecnología … o mejor dicho, una relación con el mundo a través de la tecnología») y el conocimiento, por una parte, y una ética de cooperación, autonomía e implicación con determinadas libertades, por otra. Este carácter reflexivo de su práctica los hace muy interesantes

para la investigación antropológica. El *hacking* tiene que ver más con cómo nos relacionamos con nuestras prácticas tecnológicas que con el uso con la tecnología misma. La cultura *hacker* tiene que ver fundamentalmente con esa dimensión de la cultura que consiste no sólo en seguir reglas, sino en relacionarse con ellas:

> La cultura no consiste sólo en las reglas para llevar a cabo una acción, sino también en el conjunto de *relaciones regladas* que los agentes mantienen con esas reglas, el conjunto de *disposiciones* que ponen en juego al interpretarlas.
>
> [...]
>
> Estas nuevas relaciones son inteligibles como un conjunto de *reglas de apreciación, reconocimiento o interpretación* de las reglas del juego social (Díaz de Rada, 2010, págs. 187-188).

Es decir, tiene que ver con la dimensión de reflexión y evaluación de las reglas, y por lo tanto con la ética de las prácticas. Y es que hoy se piensa la figura del *hacker* a partir de lo que se ha llamado la *ética hacker*. En *Hackers. Heroes of the Computer Revolution*, Levy resume los principios fundamentales de está ética: el acceso a los ordenadores debe ser ilimitado y total; toda la información debe ser libre; desconfía de la autoridad y promueve la descentralización; los *hackers* deben ser juzgados por su actividad de *hacking*, no por criterios irrelevantes como títulos, edad, raza o posición; puedes crear arte y belleza con un ordenador; y los ordenadores pueden mejorar tu vida (Levy, 1984, págs. 39-49).[6] Ésta es la formulación que ha hecho más fortuna, pero otras referencias importantes para entender la ética y la cultura *hacker* son Himanen (2003), Coleman y Golub (2008), Thomas (2002), Clemmitt (2011), y sobre todo Coleman (2013).

La relación de los miembros de Debian con la figura del *hacker* no es siempre clara ni unívoca pero sí está siempre presente (véase la sección 1.1, *Una visión preliminar. Tres escenas etnográficas*). Como veremos en el capítulo 6, *Debian*

[6] Una crítica muy interesante de esta formulación de la ética *hacker* realizada a partir de la propia cultura *hacker* se puede ver en una conferencia de Allison Parrish en la *Open Hardware Summit* de 2016 (Parrish, 2016).

como comunidad y como colectivo. ¿Quién es miembro de Debian?, los principales espacios en las reuniones del Proyecto, las DebConfs, se denominan *hacklabs*, y ellos usan habitualmente el término *hacking* para referirse a su actividad. Los Desarrolladores de Debian (y de *software* libre en general) comparten con los *hackers* el gusto por la tecnología, pero también la conciencia de la imbricación de política y tecnología. La figura del *hacker* es un elemento importante de la caracterización de un determinado «régimen de vida» o forma de vida colectiva constituida a través de prácticas reflexivas que son a la vez tecnológicas, éticas y políticas (Ong y Collier, 2005b). En esta forma de vida se entrecruzan muchos elementos que caracterizan nuestro campo de estudio: la relevancia de los dispositivos tecnológicos para comprender la formación de los colectivos; las prácticas del compartir y la cooperación; los conflictos en torno al conocimiento y las formas de propiedad intelectual; la meritocracia y la doocracia; o la rentabilización como capital simbólico del esfuerzo que va más allá del marco laboral y el interés porque la propia actividad no dependa de las necesidades de una empresa (Wolf, 2011b). La figura del *hacker* ilustra la reunión provisional y variable en una identidad subjetiva de una gran variedad de elementos heterogéneos. Es un actor que destaca en la capacidad de modificar redes y asociaciones ya establecidas.

Así pues, la del *hacker*, como la del *bug*, es una figura que refuta en sus prácticas el determinismo tecnológico que pone el acento en la determinación de las relaciones sociales por los dispositivos tecnológicos. Pero también el determinismo social que enfatiza las intenciones de los individuos y entiende los artefactos técnicos como simples intermediarios. Ni unos ni otros existen aisladamente en el campo de prácticas en el que actúan. No es posible dar cuenta de la agencia de unos sin atender a la de los otros, puesto que los actores relevantes lo son a partir de su asociación. Muestra claramente como en su acción social no se puede partir *a priori* de una distinción entre elementos técnicos y sociales, instrumentales y expresivos. En su acción, que sólo puede ser acción social, aparecen siempre entrelazados. Sólo podemos distinguirlos *a posteriori*.

Ambas figuras, la del *bug* y la del *hacker*, remiten a y aclaran aunque de manera diferente la noción del código como actante. Si el *bug* pone de manifiesto el carácter de mediador del código, el *hacker* es un experto en producir esas diferencias inesperadas, esas traducciones que caracterizan a los mediadores. Es un

experto en contraponer su propio antiprograma a los programas de acción con los que se encuentra. Por otra parte, el *bug*, incluso la propia característica de ser un *bug*, sólo se puede dar en agenciamientos concretos. El *hacker* es precisamente también un experto en agenciamientos.

4.3 Instrumentalidad y expresividad del código.

A partir de la distinción que veíamos al principio de este capítulo entre código fuente y código binario, se sigue que una de las características del código fuente más importante es el hecho de que no es sólo una herramienta, un instrumento funcional, sino una forma de expresión. Recordemos una cita que vimos en la introducción:

> **gwolf:** Me gusta definir el [código] fuente como vehículo de comunicación entre humanos en un lenguaje formal – tan especializado como las partituras, fórmulas u otros.

En el extremo podemos encontrar la concepción de la programación de uno de los teóricos más prestigiosos e influyentes de las ciencias de la computación, la *literate programming* (Literateprogramming.com, s.f.):

> I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title: «Literate Programming.»
>
> Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.
>
> The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style.

> Such an author, with thesaurus in hand, chooses the names of variables carefully and explains what each variable means. He or she strives for a program that is comprehensible because its concepts have been introduced in an order that is best for human understanding, using a mixture of formal and informal methods that reinforce each other (Knuth, 1984, pág. 97).

La escritura y modificación del código fuente es así una práctica en la que aparecen siempre elementos instrumentales y expresivos, y en la que ambos son inseparables. Posiblemente podríamos afirmar esto mismo de cualquier tipo de herramienta, pero de nuevo nos encontramos aquí que los objetos informáticos muestran esto más claramente, debido a las características propias del *software*.

El código es expresivo en primer lugar porque de hecho programar es literalmente producir signos. El programador individual, o el colectivo que programa, se expresa pues a través del código. Pero como vimos en la sección 2.2.8, *Agencia y semiótica*, producir un signo es una de las dimensiones de la agencia residencial. Ahora bien, la agencia, como vimos, es siempre gradual y distribuida, y tiene lugar siempre en agenciamientos. El código no se limita a transportar sin modificar las intenciones de quien lo escribe, sino que las traduce, es decir, las modifica (véase la sección 2.1.7, *Procesos de traducción*). El código traduce al programador, pero también, por ejemplo, la licencia (véase la sección 5.2.1, *Las licencias libres*) traduce al programa al que se aplica, así como a la misma ley de *copyright*. Es decir, el código expresa porque es un mediador y no un simple intermediario, en tanto modifica los fines de los demás. En este sentido, el código tiene una dimensión expresiva porque efectivamente expresa y traduce (modifica) fines, pero esto no quiere decir que sea solamente una expresión del programador. Lo que se expresa es también el colectivo sociotécnico y los agenciamientos implicados, en tanto que se produce un reparto de la agencia a lo largo de todo el entramado sociotécnico. Por todo esto, una distinción rígida entre instrumentalidad y expresividad deja de tener sentido. Entenderemos pues que son dos dimensiones de las acciones, inseparablemente unidas.

Los elementos expresivos de la escritura y modificación del código, en rela-

ción a los *hackers* y desarrolladores de *software* libre, han sido bien establecidos y discutidos por Coleman (2005, 2009, 2013). El código es, además de una herramienta que tiene una determinada funcionalidad, una forma de discurso expresivo y una forma de conocimiento social. Ahora bien, el conocimiento y su expresión están siempre ligados a controversias de orden político, aunque sólo sea porque existen restricciones para su distribución, especialmente del código fuente. Coleman explica con mucha claridad la relación entre acceso al conocimiento y política:

> although hacking is often perceived as apolitical, hacking always tends to evoke political elements due to the nature of knowledge in our society. The quest for knowledge, which is an unmistakable core component of hacking, is a politics of transgression because the 'knowledge' that is sought is often inaccessible (or potentially so) at either a technological or legal reason. It is this political condition that helps to explain the initial entry into the 'ethical realm' for hackers (Coleman, 2003, pág. 297).

Una importante consecuencia de que el *software* sea una forma de expresión es que, aunque lo pone bajo la legislación del *copyright*, también le asegura las protecciones legales a la libertad de expresión. Se trata de una expresividad tanto poética como política. Sobre esta cuestión, Coleman (2009, 2013) ofrece un análisis histórico y conceptual muy detallado. También en Kelty encontramos la noción de que la escritura de código se sitúa entre una práctica técnica y una práctica expresiva:

> source code [...] it is both an expressive medium, like writing or speech, and a tool that performs concrete actions (Kelty, 2005a, pág. 119).
>
> [...]
>
> the practice of writing software is precariously situated between verbal argument and material practice; indeed, software creation itself represents a certain immanent critique of the very distinction between speech and practice (Kelty, 2005a, pág. 186).

Y Latour llega a generalizar la existencia de elementos expresivos, a partir del código y los ordenadores, a otros tipos de entidades materiales: «Thanks to computers, we now know that there are only differences of degree between matter and texts» (Latour, 1996, pág. 22).

Esta dimensión expresiva recoge entonces elementos de creatividad poética y estética, que aquí no consideraremos,[7] pero sobre todo elementos éticos, sociales y políticos. Estos elementos no son independientes entre sí: la apreciación estética del código está ligada a la posibilidad de su reutilización y extensión posterior, así como a la posibilidad de ser fácilmente modificable, lo que tiene un componente social y político esencial, fundamental para el *software* libre. Leach *et al.* (2009, pág. 51) lo expresan claramente refiriéndose a los desarrolladores de *software* libre en general:

> the moral imagination observable in this phenomenon can be understood with reference to its emergence around specific methods of technical production. Principles of openness, truth, freedom and progress, which are also understood as central to the technical production of good software, are reinforced (as a ethical orientation) by their contribution to making 'good' software.

Hacer «buen» *software* en sentido técnico pero al mismo tiempo en sentido moral implica entonces expresar determinados principios de diseño en el código.[8] Estos principios sociales, morales y estéticos no son algo añadido a un núcleo de actividad técnica, que podrían modificarla externamente en uno u otro sentido. Son parte intrínseca de esa misma actividad, y su realización depende precisamente de ella. La actividad de construir dispositivos tecnológicos es en sí misma ética y política, como vimos también en la sección 3.3, *Debian como público recursivo*:

[7] Al respecto puede verse Dexter *et al.* (2011).

[8] Aunque esta cuestión excede con mucho las posibilidades de la presente investigación, se podrían considerar las consecuencias sociales y políticas que introducen en la escritura del código diferentes estilos de programación, o el uso de diferentes lenguajes con diferentes potencias y capacidades expresivas.

> For here we see a clear example of a social creation in a holistic sense. Morality and ethics are inseparable from the objects which the community produces. Politics, and an imagined future, are pursued through the construction and development of software, of objects, which themselves are to carry the burden of, and are believed to instantiate, an ethical and moral vision. Politics in this case can be engaged through writing software in a particular manner (Leach *et al.* 2009, pág. 66).

Y esta dimensión política tiene que ver con el hecho de que el código es una forma de conocimiento y expresión sociales. Un práctica esencial para aprender a programar es la lectura de código que otros han escrito. Este aprendizaje es un proceso social, que requiere del acceso al código fuente. MacKenzie cita una entrevista con DeMillo, uno de los autores de un influyente artículo (DeMillo *et al.* 1977) sobre dos formas diferentes de prueba matemática, la realizada efectivamente por matemáticos y la obtenida a través de programas informáticos:

> Being unreadable and –literally– unspeakable, verifications cannot be internalized, transformed, generalized, used, connected to other disciplines, and eventually incorporated into a community consciousness. They cannot acquire credibility gradually, as a mathematical theorem does; one either believes them blindly, as a pure act of faith, or not at all (entrevista con DeMillo, citado por MacKenzie (2004, pág. 207)).

Aquí, «verificaciones» se refiere a las pruebas generadas por ordenador. Lo que les falta es el proceso social de verificación, el proceso por el que la prueba llegada a ser creída e incorporada por la comunidad científica. Un proceso que incluye ser leído, evaluado, publicado, discutido, internalizado, parafraseado, generalizado, usado, y conectado con otros teoremas (MacKenzie, 2004, págs. 203-204). Nada de esto es posible hacerlo con una prueba generada por ordenador, porque de hecho es ilegible. No cumplen por lo tanto con su objetivo instrumental. Vemos aquí una analogía clara con la diferencia entre el código fuente y el código binario, producido por un compilador. El acceso al código fuente posibilita el proceso social de creación de conocimiento que se expresa en el *software*.

¿Cómo podemos entender y mostrar la relación, entonces, entre la dimensión instrumental, funcional, de esta práctica social y su dimensión expresiva? Ya hemos insistido en que a partir de una concepción semiótica de la agencia son inseparables (véase la sección 2.2.8, *Agencia y semiótica*). Ratto (2005b) expresa una idea similar al presentar Linux (en este caso se refiere al *kernel*, el núcleo del sistema operativo) como un artefacto que es, a la vez, un objeto retórico:

> Linux, then, makes obvious what is equally true of all human-made artifacts. It exists as a technical object that is used both in materially productive tasks (i.e., to operate computer hardware) and in socially productive work (i.e., to make claims about labour and organization) (Ratto, 2005b, pág. 207).

Es decir, es un objeto retórico porque contribuye a crear significados socialmente productivos. Pero a la vez, nos advierte precisamente del peligro de separar los elementos expresivos y los funcionales del *software*. El problema consistiría sobre todo en reservar una explicación «social» para los primeros (representados históricamente sobre todo en el estudio de las *interfaces*), como un añadido independiente de una explicación puramente racional o técnica de los segundos. El *software* es expresivo, pero esa expresividad está inserta y es inseparable de la estructura técnica, instrumental, del código. Para poner esto de manifiesto, Ratto usa la expresión «embedded technical expression of software»:[9] «that is, how software programs express normative positions about users, programmers, appropriate tasks, and social organization» (Ratto, 2005b, pág. 207).

Ratto propone un esquema señalando tres aspectos en los que se muestra esta expresividad del código inserta en sus características técnicas. El primero es el hecho de que el código fuente expresa también buenas prácticas de programación, que aquí no discutiremos. El segundo es el comportamiento del programa, que permite, facilita o prohíbe las acciones de los usuarios, expresando por tanto cuál es el comportamiento esperado y apropiado. Esto supone, como ya hemos visto

[9] Kelty usa una expresión similar: «Openness and scalability, modularity and trust, transparency and security; [...] are no less terms of political contest for being embedded in technical practice» (Kelty, 2005a, pág. 185).

antes, que el programa es un agente, un mediador, y que inscribe un determinado programa de acción.

Veamos un ejemplo simple. Entre finales de mayo y principios de junio de 2009, se produjo una discusión en torno al hecho de que en el lector de archivos PDF `okular` del entorno de escritorio KDE, incluido en la distribución Debian, esté activada por defecto una opción que hace que el programa obedezca las restricciones relativas al DRM (*Digital Rights Management*), y que impide por ejemplo copiar partes del texto. Esta es una discusión interesante que afecta a varias cuestiones: la gestión de derechos de autor en el entorno digital y las concepciones de lo que es un sistema libre y abierto; la distinción entre cuestiones legales y técnicas; la coordinación entre diferentes mantenedores; la relación de Debian con los desarrolladores *upstream* (los que desarrollan el programa, que luego es empaquetado y distribuido por las diferentes distribuciones), o la autonomía de los Desarrolladores de Debian. Pero considerémoslo desde la perspectiva de como afecta al «comportamiento esperado y apropiado» por parte de los usuarios.

El 31 de mayo de 2009, un Desarrollador de Debian escribió el siguiente *post* en su *blog* (Goerzen, 2009):

> So I just recently switched to KDE 4 (still using it with xmonad, of course) and I just now ran into my first reall big annoyance.
>
> I just downloaded a PDF, and tried to copy and paste a bit of text from it. I used the selection tool, and Okular (KDE's document viewer) offered to speak it to me, but said "Copy forbidden by DRM."
>
> pdftotext was able to convert the entire file to text format in an instant.
>
> Why are people intentionally adding code to KDE to remove my freedom? This is crazy and nuts. Nobody should be doing this, least of all in Free Software!

Se puede seguir la discusión y sus argumentos en la lista de correo `debian-devel`[10] y en el BTS. Uno de los mensajes de esta discusión es el siguiente:

[10] A partir de Debian-devel (2009).

```
To: John Goerzen <jgoerzen@complete.org>, 531221@bugs.debian.org
Cc: debian-devel@lists.debian.org, Pino Toscano <pino@kde.org>
Subject: Re: Bug#531221: okular: Arbitrarily enforces DRM
From: Adeodato Simó <dato@net.com.org.es>
Date: Sun, 31 May 2009 02:30:58 +0100

> I'm CCing this to Debian-devel because I think it speaks to a larger
> issue.
> I just downloaded a PDF, and tried to copy and paste a bit of text
> from it. I used the selection tool, and Okular offered to speak it to
> me, but said «Copy forbidden by DRM.»
> pdftotext was able to convert the entire file to text format in an
> instant.
> So what I want to know is: why are people putting code into Debian
> that limits our freedom? Why are people putting such code into KDE?
> And can we please patch it to stop that?

I see it's been pointed out in a comment in your blog post already, but
I'll mention it here for the benefit of those reading along: obeying DRM
is a configurable runtime option in Okular, so it's just a matter of going
to the preferences dialog and unchecking the «Obey DRM» check box.

Now I have no idea why it would default to obeying it (or, for that matter,
why it would have such an option). I'm CC'ing Pino whom I'm sure will be
able to help. (My guess would be that it protects upstream against some
shit or whatever, at least by their reckoning, or the person that added
it in the first place.)

Cheers,
– - Are you sure we're good? - Always. – Rory and Lorelai
```

Como señala este mensaje, basta con escoger una de las opciones del programa para que éste no obedezca a las restricciones y sea posible cambiar el texto. Pero la discusión sigue más allá de esto, en torno a la cuestión de si el programa debe ofrecer la opción de activar la opción de respetar las restricciones, o en todo caso cuál debe ser la opción por defecto. Uno de los argumentos más relevantes es lo que los usuarios van a entender como comportamiento posible e incluso deseable según se implementen estas opciones, si van a dar por supuesto que es legal copiar el archivo, o si por el contrario se les va a llevar a obedecer una restricción injusta.

Son decisiones que se están tomando en el código,[11] de tal modo que su diseño afecta a la agencia y los fines de los usuarios. Pero también se está decidiendo sobre la distribución de la agencia, sobre quién debe determinar estas cuestiones: ¿el estándar que describe los documentos PDF? ¿El desarrollador *upstream* (véase la sección 4.6.1, Debian como *Proxy*. *Upstream* y *downstream*) de `okular`? ¿El mantenedor del paquete en Debian? ¿Los usuarios?

Pero el aspecto más interesante de la expresividad del código que describe Ratto es el tercero, que se centra en la estructura del código:

> I define *code structure* as the organization of different code elements within the overall software program. [...] I see code structure as expressive of the social organization of the act of programming (Ratto, 2005b, pág. 209).

La estructura del código organiza la funcionalidad del código. Pero indisociablemente organiza y expresa también las relaciones entre los programadores que trabajan sobre ese código. En Debian, esto se ve claramente en la estructura de los paquetes que componen la distribución y en el proceso de su preparación (lo veremos en la sección 4.5, *El empaquetado como práctica sociotécnica*). Para ejemplificar esta cuestión, Ratto se refiere al cambio que se produjo con la versión 2.0 del *kernel* Linux, en 1995. En esa versión se introdujo la arquitectura que permitía el uso de módulos cargables del núcleo (o simplemente módulos del núcleo). Éstos son partes del núcleo (por ejemplo, controladores de dispositivos o de sistemas de ficheros) que se pueden cargar (y descargar) según sean necesarios, y mientras el núcleo está funcionando, extendiendo así su funcionalidad. Además de eliminar la necesidad de recompilar el núcleo con frecuencia, facilitar su mantenimiento y la corrección de errores, y ahorrar memoria (Tldp.org, s.f.), esto permitía el trabajo en diferentes módulos por parte de los desarrolladores sin tener que coordinarse previamente, así como la posibilidad de usar módulos que no tuviesen que ser aprobados previamente por Linus Torvalds, e incluso elimi-

[11] Ésta es una de las tesis del influyente libro de Lawrence Lessig *El código 2.0* (Lessig, 2009).

naba la exigencia de que estuviesen licenciados con la GPL.[12] Es un diseño que, al mismo tiempo, cambia la estructura del código y los requerimientos técnicos de su desarrollo, y las formas de organización de los desarrolladores, e incluso influye en los aspectos legales relativos a las licencias. En palabras del propio Linus Torvalds:

> So once again managing people and managing code led to the same design decision. To keep the number of people working on Linux coordinated, we needed something like kernel modules. But from a design point of view, it was also the right thing to do (Torvalds (1999), citado en Ratto (2005b, pág. 210)).

Curiosamente, en otro lugar de ese mismo texto Torvalds hace una distinción entre los problemas técnicos y los sociales muy similar a lo que vimos en la sección 2.2.1, *Concepciones* emic *de lo técnico y lo social* sobre esa distinción en Debian:

> The other part of modularity is less obvious, and more problematic. This is the run-time loading part, which everyone agrees is a good thing, but leads to new problems. The first problem is technical, but technical problems are (almost) always the easiest to solve. The more important problem is the non-technical issues. For example, at which point is a module a derived work of Linux, and therefore under the GPL? (Torvalds, 1999, pág. 56).

En cualquier caso, una idea que expresa aquí Torvalds, que hemos visto ya varias veces y que detallaremos en el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*, es la necesidad de coordinar simultáneamente a personas y objetos (código en este caso). Aquí está la clave de la noción de colectivo sociotécnico.

[12] Para un estudio de cómo Linux está diseñado para ser fácilmente rediseñado por otros usuarios y desarrolladores, véase también Ratto (2003).

Así, la expresividad del código está inscrita en, y es inseparable de, su funcionalidad, dado que ésta contribuye a organizar el conjunto de relaciones entre personas y artefactos que constituyen el colectivo. Ratto afirma así que el código es doblemente expresivo: «software as code, behavior, and structure both *expresses* relationships and *organizes* those relationships» (Ratto, 2005b, pág. 211). Lo veíamos también a propósito de la organización de los usuarios y grupos de UNIX y GNU/Linux en relación con los privilegios de los miembros de Debian en la sección 4.1, *La Red como metáfora. Unix*, así como en la sección 3.3.2, *Apertura y libertad*, a propósito del artículo de Kahn Gillmor (2007).

4.4 El código como objeto híbrido.

¿Qué tipo de entidad son entonces el *software* o el código? Dada su complejidad y sus múltiples dimensiones, podríamos considerar el *software* distribuido por Debian como un *híbrido* o cuasi-objeto según la caracterización de Latour: «No bien seguimos de cerca algún casi-objeto, se nos aparece a veces como cosa, a veces como relato, a veces como lazo social, sin reducirse jamás a un simple ente» (Latour, 2007, pág. 131). Éstas son características tanto de las redes como de los cuasi-objetos:

> las redes [...] son reales, colectivas y discursivas [...] ¿Es nuestra la culpa si *las redes son a la vez reales como la naturaleza, narradas como el discurso, colectivas como la sociedad*? (Latour, 2007, pág. 22).
>
> Reales como la naturaleza, narrados como el discurso, colectivos como la sociedad, existenciales como el Ser, tales son los cuasi-objetos que los modernos hicieron proliferar, y así conviene seguirlos, volviendo a ser simplemente lo que jamás dejamos de ser, no modernos (Latour, 2007, pág. 133).

Los objetos híbridos son entonces realidades (cuasi-objetos y cuasi-sujetos) que se sitúan entre lo social y lo natural, entre lo humano y lo no-humano. Lo que los caracteriza es esa multidimensionalidad. En *Aramis or the love of technology*, Latour muestra como los ordenadores (y aquí se debe entender este término de

manera genérica, sin distinguir entre *hardware* y *software*) son un ejemplo de este tipo de realidad:

> Let's reinscribe in them [microprocessors] the entire set of action programs that we can no longer take care of by legal, social, or traditional means. [...] They hold everything. [...] Where is this being, the microprocessor, to be situated? On the side of human beings? No, since humans have delegated, transcribed, inscribed their qualities into nonhumans. On the side of nonhumans, then? Not there either. [...] Thus, the object, the real thing, the thing that acts, exists only provided that it *holds humans and nonhumans together, continuously*.
>
> [...] For the thing we are looking for is not a human thing, nor is it an inhuman thing. It offers, rather, a continuous passage, a commerce, an *interchange*, between what humans inscribe in it and what it prescribes to humans. It translates the one into the other. [...] What should it be called? Neither object nor subject. An instituted object, quasi-object, quasi-subject (Latour, 1996, págs. 212-213).

Esta cita es muy extensa pero también muy ilustrativa, ya que nos muestra con claridad en qué sentido los dispositivos tecnológicos informáticos son objetos híbridos. Como venimos detallando y hemos visto en numerosos ejemplos, no es posible describir una de estas dimensiones sin que nos veamos llevados a otra. Pero lo que es fundamental es que esta doble dimensión de los objetos híbridos no es resultado de una combinación de realidades diferentes, sino por el contrario el resultado de un proceso de separación:

> No necesitamos enganchar nuestras explicaciones a esas dos formas puras, el objeto o el sujeto-sociedad, porque son ellas, por el contrario, las que son resultados parciales y purificados de la práctica central, única que nos interesa. Son el producto del *cracking* purificador y no su materia prima. En efecto, la naturaleza gira, pero no alrededor del sujeto-sociedad. Gira alrededor del colectivo productor de cosas y

> de hombres. En efecto, el sujeto gira, pero no alrededor de la naturaleza. Es obtenido a partir del colectivo productor de hombres y cosas (Latour, 2007, pág. 118).

En nuestro campo, más que entre lo natural y lo social nos encontramos con que la distinción relevante es la que se produce entre lo técnico y lo social. Esta perspectiva nos permite entonces abordar el estudio de la distinción establecida entre lo «técnico» y lo «social» como producto de un proceso, de un trabajo de separación entre ambos polos, como vimos en la sección 2.2.3, *La producción práctica de la distinción*. De un proceso de *purificación* (Latour, 2007) que hace que estos cuasi-objetos se desplacen, en la dimensión técnico-social, hacia el polo de lo técnico. Se entiende entonces, como vimos, la preeminencia de esta dimensión, y más significativamente, la definición de lo «social» como el resto que queda, una vez que se ha realizado ese proceso de separación.

De lo que hemos intentado partir en este trabajo es de los múltiples objetos híbridos que proliferan en el ámbito intermedio entre diversos dualismos: lo técnico y lo social, las herramientas y el conocimiento, lo instrumental y lo expresivo, o (como veremos en el capítulo 5, *El don del* software *libre*), entre el don y la mercancía, también entre lo que se dona y lo que se guarda. Y partir del hecho precisamente de que son híbridos, de que todos esos diferentes aspectos no son elementos que se añaden externamente a un código puramente técnico o instrumental. El código es ya todo eso. Si se puede afirmar que lo social y lo técnico son objetos que se producen, es porque hay objetos híbridos a partir de los que se pueden producir. Esta separación es un proceso de purificación que sólo tiene sentido si hay objetos híbridos para empezar.

Si un Desarrollador crea un programa, puede elegir distribuirlo con una licencia libre o no (véase la sección 5.2.1, *Las licencias libres*). Así, parece que el hecho de que sea *libre* es algo añadido externamente al funcionamiento técnico del programa. Pero en muchas ocasiones el hecho de que se vaya a licenciar con una licencia libre transforma la naturaleza del código que se escribe (porque se reutilicen partes de otros programas, para facilitar la contribución externa, porque el código va a ser examinado por otros, ...). Ese código es un híbrido desde

su origen. En la actividad normal de un programador, estas cuestiones sociales, legales y técnicas forman parte del diseño de su código.

El proceso de separación es un proceso central en el colectivo que estamos estudiando porque, como hemos visto, es una condición para determinar cómo se distribuye la agencia en el colectivo, cómo se ejerce la doocracia. Trasladar una controversia al campo de lo técnico es una condición para la existencia y aplicación de la doocracia. La doocracia funciona cuando una cuestión se transforma en una cuestión técnica, cuando una solución se puede inscribir como programa de acción en un dispositivo tecnológico.

Hemos estado considerando al código como el objeto en el que se producen estas hibridaciones, estas mediaciones o traducciones. Pero el mismo código es ya el producto de un proceso de separación entre *hardware* y *software*, como vimos en la sección 4.1, *La Red como metáfora. Unix*. En esta historia, el *software* libre y abierto profundiza de hecho esa separación, y precisamente (como seguiremos viendo en la sección 5.1.1, *Código fuente y don*) para poder ser donado y compartido.

Volvamos por un momento a la discusión sobre el *firmware* no libre que examinamos en la sección 2.2.4, *Un ejemplo: el* Firmware *y la inscripción de programas de acción*. Allí veíamos que una de las controversias al respecto era la de si el *firmware* era o no una forma de *software*. Si el *firmware* es una entidad que se encuentra entre el *hardware* y el *software* (un objeto híbrido), lo que está aquí en juego es precisamente hacia cuál de los dos polos se desplaza. La solución de Debian pasó por llevarla hacia el campo del *software*, en contribuir a la separación del *software* respecto al *hardware*, incluyendo (al menos acercando) al *firmware* en el primero. Veamos un *post* del DPL de aquél momento en el que anuncia la decisión (Zacchiroli, 2010c):[13]

> Today we have announced that, starting with the upcoming release of Squeeze, Debian will be even Free-er. Exceptions to the DFSG for non-free Linux firmware blobs, which have been granted in the past, will

[13] Cito sólo el inicio del *post*, pero en el mismo aparecen más cuestiones interesantes relativas a esta controversia, algunas de las cuales vimos en la sección correspondiente.

> no longer be granted. Starting from Squeeze, Debian will be Free the bottom up, no matter where your own definition of software ends.

«No matter where your own definition of software ends», es decir, aun incluyendo el *firmware*. Porque de hecho se le puede tratar como tal, al haberlo separado. Y por lo tanto puede o no ser libre. A lo largo de toda esta controversia, los componentes legales y éticos han contribuido a definir lo que es *software* y a ordenar el archivo de Debian. La definición de lo que es *software*, la especificación de qué significa *libre*, la determinación de quién puede decidir estas cuestiones, y todos los elementos que hemos estado examinando, se anudan en torno a ese objeto complejo que es el código, de tal modo que cada elemento modifica a los demás. Por eso decíamos en el capítulo anterior que sólo se puede entender la doocracia en Debian teniendo en cuenta la naturaleza del *software*.

Vamos a intentar a continuación detallar algunas de las maneras en que los miembros de Debian se relacionan en sus prácticas cotidianas con el código.

4.5 El empaquetado como práctica sociotécnica.

A estas alturas resultará ya claro que las relaciones que los miembros de Debian establecen con el código son múltiples, variadas y complejas. No se puede realizar pues un análisis unívoco que tenga esta relación como objeto aislado. Más bien, es este conjunto de relaciones lo que se usa para analizar la constitución de la constelación de fenómenos sociotécnicos que consideramos. Sin embargo, podemos distinguir una práctica social en relación al código que ocupa un lugar central en este campo social: el empaquetado de *software*, su preparación para formar parte de la distribución Debian.

Como veremos en el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*, no todos los miembros de Debian se dedican a empaquetar o mantener paquetes. Y desde luego no es necesario ser miembro formal del Proyecto para hacerlo, como también veremos en ese mismo capítulo, aunque al principio ambas cosas estaban mucho más estrechamente unidas. Aun

así, sigue siendo la actividad que define en mayor medida, y la que dota de sentido, a lo que se hace en Debian. Porque lo que ofrecen a fin de cuentas es una distribución, es decir, una colección de paquetes de *software* (véase el cuadro 1.7, *Distribuciones GNU/Linux*). Como decía un Desarrollador de Debian:

> **bencer:** Empaquetar no es desarrollar del todo, es más bien integrar la aplicación en el resto del sistema. Es más integración que desarrollo. Nos debían llamar más integradores que desarrolladores. También hay desarrollo, pero creo que es la parte menor.

Además, entender qué es un paquete de *software* será fundamental para entender qué significa contribuir y el don del *software*, como veremos en el capítulo siguiente. Lo que Debian contribuye a la comunidad del *software* libre consiste precisamente en esto.

Recordemos brevemente lo que se dijo en la sección 2.1, *Bug #573745*, a propósito del empaquetado de `python`. Un paquete de *software* es el conjunto de archivos que implementan una determinada funcionalidad o característica en un sistema informático. Esta funcionalidad puede ser un programa, una biblioteca, documentación, etc. La labor del empaquetador (o mantenedor, si el paquete en cuestión forma parte de la distribución oficial) es la de realizar las modificaciones necesarias en el código fuente original (o *upstream*) del *software* en cuestión de tal modo que se posibilite su integración en la distribución.

Así pues, lo que se crea en primer lugar a partir de esta labor es un paquete de *software*, binario o fuente, como vimos en la sección 2.1, *Bug #573745*, y que podrá ser instalado y manejado con las herramientas de la distribución (véase la figura 4.1, *Algunos conceptos y términos sobre paquetes*), como `apt` o `dpkg`.

Un paquete binario (Wiki.debian.org, s.f.[r]) (un fichero con extensión `.deb`) consiste entonces en el conjunto de archivos que es necesario instalar en el sistema, pero incluye también todos los datos e información necesarios para permitir su integración en el sistema en el que se instala, como de qué paquetes depende o con cuáles es incompatible. Esta información permite también la actualización

.deb	La extensión de archivo que identifica los paquetes binarios.
.dsc	La extensión de archivo que contiene la descripción de un paquete fuente y su composición (Hertzog y Mas, 2012, pág. 86).
dpkg	El programa, desarrollado dentro de Debian, que maneja los archivos binarios con extensión `.deb`.
apt	Conjunto de programas que permite la ejecución de modificaciones de alto nivel en el sistema: instalar o desinstalar un paquete (manteniendo las dependencias satisfechas), actualizar el sistema, listar los paquetes disponibles, etc. (Hertzog y Mas, 2012, pág. 74).

Cuadro 4.1: Algunos conceptos y términos sobre paquetes

ordenada de un gran número de programas. El principal archivo que contiene esta información se denomina `control`.

Por su parte, un paquete fuente (Wiki.debian.org, s.f.[s]) se compone de tres elementos: el código fuente original del *software*, usualmente desarrollado fuera de Debian; el conjunto de cambios que se aplican a ese código para su inclusión en Debian; y el archivo de texto con extensión `.dsc` (`Debian Source Control`) que lo describe e indica los archivos que lo componen. Como ejemplo, el archivo `org-mode_7.9.2-1.dsc` que describe el paquete fuente `org-mode`[14] contiene lo siguiente:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Format: 3.0 (quilt)
Source: org-mode
Binary: org-mode
Architecture: all
Version: 7.9.2-1
Maintainer: Sebastien Delafond <seb@debian.org>
Homepage: http://orgmode.org
Standards-Version: 3.9.3
```

[14] Usado, entre otras cosas, para escribir esta tesis doctoral.

```
Build-Depends: debhelper (>= 6)
Build-Depends-Indep: texinfo, texlive-latex-base, emacs | emacs24 | emacs23
Package-List:
org-mode deb misc optional
Checksums-Sha1:
4671c0f2b93b9c208e286331433d4baec792d5e1 2605478 org-mode_7.9.2.orig.tar.gz
3364017e37f9949df1cfb3fe3fb4eb341ed80b2b 7889 org-mode_7.9.2-1.debian.tar.gz
Checksums-Sha256:
320334cfca1d1581491d9dd9ad66f99cf2bfe7abcd475b9bc4ad9e3a3712a01b 2605478
    org-mode_7.9.2.orig.tar.gz
acf44bbdab7b4d89995dc614adf3906b7e70562222ee2da6fe1caa05e9a79619 7889
    org-mode_7.9.2-1.debian.tar.gz
Files:
242f942fc600c485c9a8db2b1b235644 2605478 org-mode_7.9.2.orig.tar.gz
7d3ab2c02afea2a79c21be91694229a1 7889 org-mode_7.9.2-1.debian.tar.gz

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.11 (GNU/Linux)

iEYEARECAAYFAlCPjHIACgkQiZgNKcDdyD8UkgCgvLN6x2C68QBdWo6AlESdKEyF
rDcAoJSHDVEN9a0ID6/TiD2K9IY/BAIf
=bATM

-----END PGP SIGNATURE-----
```

Este archivo nos indica varias cosas. El nombre del archivo binario que se producirá, la versión del programa, de qué otros programas depende en el momento de la compilación (la construcción del binario), quién mantiene el paquete en Debian, la página del proyecto original o *upstream*, o los archivos que componen el paquete fuente. Obsérvese también que aparece el *checksum* (suma de verificación) que permite comprobar la integridad de las fuentes, y que el archivo ha sido firmado digitalmente con una clave GPG, asegurando así su autenticidad y que no ha sido manipulado.[15]

[15] Véase la sección 6.4.1, *Identidad y claves GPG*.

Los paquetes fuente son entonces clave para entender cualquier proceso sociotécnico en Debian. Lo que Debian distribuye, su distribución, surge de una colección de paquetes fuente. Toda acción social se articula en torno a su producción y mantenimiento. Incluso aunque un usuario sólo se interese por los binarios, la contribución de los miembros de Debian y su trabajo se materializa en los primeros. En palabras de los Desarrolladores de Debian Raphaël Hertzog y Roland Mas:

> The source package is the foundation of everything in Debian. All Debian packages come from a source package, and each modification in a Debian package is the consequence of a modification made to the source package. The Debian maintainers work with the source package, knowing, however, the consequences of their actions on the binary packages. The fruits of their labors are thus found in the source packages available from Debian: you can easily go back to them and everything stems from them (Hertzog y Mas, 2012, pág. 88).

Los autores se refieren principalmente a la relación entre paquetes fuente y paquetes binarios, enfatizando el papel de los primeros en la distribución Debian. Pero podemos generalizar y poner de relieve su papel fundamental no solamente en la distribución (el *software* distribuido), sino más ampliamente en el Proyecto Debian, en tanto que efectivamente componen la materialización de la acción social que se despliega en su seno. Es aquí donde se anuda y cobra cuerpo la múltiple red de asociaciones, mediaciones y traducciones que estamos siguiendo.

¿Cómo se crea entonces un paquete binario a partir del código fuente? El grado cero de relación con el paquete de *software* sería su instalación y manejo por parte del usuario. Dadas las características del *software* libre, no es extraño pasar de ahí a su mantenimiento para la distribución, bien para introducir alguna modificación bien para empaquetarlo y tener el paquete siempre disponible más fácilmente. De hecho, ésta es una de las motivaciones frecuentes para empezar a empaquetar *software* para Debian. Veamos lo que ha de hacer un mantenedor para crear uno a partir del código fuente original. Existen varios tutoriales e intro-

ducciones para enseñar y ayudar a los novatos a empaquetar.[16] El proceso puede llegar a ser bastante complejo, pero se pone un gran interés en hacerlo accesible y en que sea fácil encontrar la información necesaria. En estos tutoriales encontramos que por una parte se introducen las herramientas que facilitarán enormemente el trabajo, automatizándolo. Por otra, se insiste en la comprensión de los conceptos subyacentes, de tal modo que siempre sea posible preparar un paquete sin su ayuda. Se enfatiza siempre la comprensión del empaquetado a bajo nivel. Las herramientas que automatizan el proceso son muy útiles, a condición de que se entienda lo que éstas hacen. Se consigue así que, aunque puedan ser consideradas como cajas negras (en las que a uno solo le interesa lo que entra y lo que sale), puedan ser siempre abiertas y examinadas. De hecho, se suele insistir en la necesidad de aprender a empaquetar observando como están construidos otros paquetes, abriéndolos y examinándolos.

El primer paso es descargar el código fuente que se quiere empaquetar, y las herramientas propias de Debian que ayudarán en esta labor. Normalmente, el código fuente del desarrollador original o *upstream* se distribuirá en un archivo comprimido que contendrá todos los archivos que lo componen (conocido como *tarball* porque se crea a partir del programa `tar`). A continuación, se renombra ese archivo para seguir las convenciones establecidas en Debian y se descomprime. Se le añaden los archivos propios de Debian en un subdirectorio (llamado precisamente `debian`) y se construye el paquete fuente de Debian, que por último será subido al archivo de Debian.

La parte principal del trabajo consiste en añadir una serie específica de archivos a ese directorio `debian`, lo que se conoce como «debianización» del paquete. Los más importantes son `debian/changelog`, donde se registran los cambios del paquete; `debian/rules`, que establece cómo se debe compilar el paquete; `debian/source/format`, que indica el formato del paquete fuente; `debian/copyright`, que incluye la información legal y la licencia del programa; y sobre todo el archivo `debian/control`, que describe el paquete. Incluye el nombre del paquete, su mantenedor, la prioridad dentro de Debian (requerido, importante, estándar, opcional o extra), a qué sección del archivo corresponde, de qué otros paquetes depende, en qué arquitecturas puede funcionar, y una descripción de su funcionalidad. Puede

[16] Véanse por ejemplo Wiki.debian.org (s.f.[t]) o Debian.org/doc (s.f.[f]).

haber, y normalmente es así, más archivos creados en el directorio `debian`.

Todos estos archivos pueden ser creados directamente por el mantenedor escribiéndolos desde cero con un editor de texto, pero se suelen usar herramientas propias de la distribución que automatizan gran parte del proceso, como `dh_make`, `debhelper` o `CDBS`, que crearán la estructura básica del paquete fuente. Una vez hecho esto, el mantenedor editará los archivos necesarios para introducir la información oportuna, y modificará el código para que funcione en Debian. En ambas tareas, se deberán seguir las especificaciones contenidas en el «Debian Policy Manual» (Debian.org/doc, s.f.[d]) y la «Debian Developer's Reference» (Debian.org/doc, s.f.[e]), que contienen los requerimientos que deben cumplir los paquetes para ser incluidos, la estructura del archivo de Debian, y los procedimientos para gestionar los *bugs*, subir paquetes, cómo hacerse cargo de un paquete o realizar un NMU (véase el cuadro 3.4, *NMU*), y diversas cuestiones sobre la organización y estructura del sistema operativo, o cuáles son las prácticas recomendadas de empaquetado.

Para un paquete sencillo el proceso puede ser muy rápido y muy automatizado. Pero empaquetar una aplicación grande y compleja puede requerir mucho esfuerzo y dedicación, además de una gran cantidad de conocimiento y habilidades.

4.5.1 Mediaciones y traducciones. Parches y cajas negras de nuevo.

Además de preparar estos archivos, el mantenedor normalmente ha de modificar el código fuente. Al distribuirse el paquete fuente con la fuente original (*upstream*) y las modificaciones introducidas para su integración en Debian por separado, cualquiera puede acceder y comprobar exactamente qué modificaciones se han introducido. Si un paquete de *software* es un ejemplar de «caja negra», las que encontramos aquí reflejan en su estructura y construcción el esfuerzo por dejarlas abiertas y accesibles.

Es decir, la construcción de un paquete fuente para Debian supone la creación de una serie de parches (véase el cuadro 2.9, *Parches*) que se aplican sobre

el código original. De hecho, el formato más usado para los paquetes fuente de Debian supone un proceso de producción de los mismos en el que ambas partes (el código fuente original por un lado, los parches para Debian por el otro) quedan nítidamente separados, y estos últimos se constituyen en la unidad básica sobre la que trabaja el mantenedor. Este formato se llama «`3.0 (quilt)`», y es lo primero que aparece en el archivo `dsc` que describe el paquete fuente, como se puede ver en el ejemplo de `org-mode`.

Como su nombre indica, este formato hace uso de `quilt`, un programa escrito para facilitar la creación, organización y aplicación de parches. En el contexto de la creación y mantenimiento de paquetes fuente para Debian, permite[17] trabajar directamente sobre el código fuente original pero manteniendo su integridad, definiendo en un archivo los parches creados, que se van almacenando en un subdirectorio determinado.

Entre las ventajas de este formato se pueden citar las siguientes:[18] el mantenimiento de parches separados y documentados adecuadamente (según la propuesta de mejora de Debian DEP-3),[19] aumentando así la facilidad de revisión y reutilización de los mismos, tanto por parte de los desarrolladores *upstream* como de distribuciones derivadas; la simplificación del proceso de trabajo de mantenimiento; y la estandarización de un único sistema de parches, lo que facilita el aprendizaje y la colaboración.

Lo interesante de estas razones y de este formato, así como de la descripción de un paquete de Debian, es que no inciden sólo en la eficiencia técnica, instrumental, del flujo de trabajo, sino que, como vimos en la sección 4.3, *Instrumentalidad y expresividad del código*, la estructura de la organización del código inscribe y articula también las relaciones entre diferentes agentes que se relacionan en su producción y circulación (el mantenedor, mantenedores de otros paquetes, revisores, desarrolladores *upstream*, distribuciones, …). Lo que Ratto denominaba *embedded technical expression of software*. La descripción del formato del paquete

[17] Para una explicación detallada, se pueden consultar Hertzog (2012a) y Wiki.debian.org (s.f.[u]).
[18] Según aparecen en Wiki.debian.org (s.f.[v]).
[19] Véase Dep-team (s.f.[c]), un documento que pretende facilitar precisamente la documentación y etiquetado adecuados para los parches, facilitando así su revisión y el trabajo colaborativo sobre ellos.

fuente es al mismo tiempo una expresión, una descripción de la comunidad que lo produce, de las relaciones que lo hacen posible. Desde la preservación del código *upstream* a los aspectos legales que se localizan en el directorio `debian/copyright`.

Esto se puede aplicar a todo el proceso de empaquetado que estamos considerando. Seguramente sería posible seguir al mantenedor desde que se sienta frente a su ordenador a construir su paquete hasta que llega al archivo de Debian, y pensar que ha actuado siguiendo una lógica puramente instrumental, técnica. Pero esta práctica sólo se puede entender en el seno de la cadena de prácticas sociales y convencionales de la que forma parte. El mantenedor no trabaja sólo sobre el *software*, ha de integrarlo en un conjunto complejo de redes. En primer lugar ha de asegurarse de que se puede integrar no sólo en su ordenador, sino en múltiples configuraciones del sistema operativo; esta compatibilidad no sólo es técnica, sino que tiene que asegurarse de que la licencia usada es compatible; de que está debidamente documentado; de que sigue las convenciones establecidas en el Proyecto; ha de aprender a trabajar con los desarrolladores *upstream* y tener en cuenta las necesidades de las distribuciones derivadas; también las de sus usuarios, siguiendo los *bugs* tanto de su paquete como de la versión *upstream*; ha de conocer el lenguaje de programación de que se trate, las herramientas de empaquetado, las bases del sistema operativo en el que se va instalar; ha de lidiar con las necesidades y la infraestructura de localización e internacionalización (véase la sección 1.2, *Origen de la investigación y acceso al campo*), por hacer una lista no exhaustiva. En definitiva, la práctica de empaquetar supone establecer toda una serie compleja de mediaciones y traducciones, estableciendo vínculos entre elementos muy heterogéneos. Es una práctica efectivamente técnica, pero en el sentido *etic* de ser un trabajo de delegación, de traducción, de inscripción de un programa de acción (véase la sección 2.2.7, *Sentidos de lo social y lo técnico*). Es decir, una práctica sociotécnica.

Y obsérvese que «empaquetar» es efectivamente casi un sinónimo de «meter en una caja negra». Y eso es precisamente, usando los términos de la Teoría del Actor-Red, lo que sucede. Se puede preparar un paquete Debian para uso personal, para instalarlo en el propio ordenador o servidor. Pero al empaquetarlo para la distribución, el mantenedor de un paquete está metiendo en él toda esa serie de relaciones y vínculos, y extendiéndolos a gran escala. Está contribuyendo a defi-

nir como pueden emplear el *software* los usuarios, entre ellos las distribuciones derivadas como Ubuntu. También a cómo se van a relacionar estos usuarios con los desarrolladores *upstream* (reenviando los *bugs* que se reporten al BTS de Debian, o limitando algunas opciones por cuestiones legales, relacionadas con las licencias). Con sus decisiones influye también en el desarrollo interno de la distribución Debian, al elegir determinadas herramientas que facilitan o dificultan la dirección que tome ésta. Se convierte así en lo que Callon (1984) llama un punto de paso obligatorio, y establece la asociación fundamental que le permitirá ejercer la doocracia (véase la sección 3.2, *Toma de decisiones: de la meritocracia a la doocracia*) e integrarse en el colectivo sociotécnico (véase la sección 6.1, *De la comunidad al colectivo sociotécnico*).

Por último, el paquete se sube al archivo de Debian, a partir del cual se distribuirá a los usuarios. Es necesario subir el paquete fuente, los binarios pueden ser construidos automáticamente en la infraestructura de Debian. Si se sube un paquete por primera vez (no como actualización de una versión anterior), deberá ser revisado antes por el equipo FTP Master para comprobar que sigue las normas, está bien construido y la licencia cumple con las *Debian Free Software Guidelines*, y que por lo tanto puede ser redistribuido por Debian. Este último proceso se explica en la sección 5.2.3, *Las licencias en Debian*.

Una vez que el paquete está en el archivo, el Desarrollador o Mantenedor se ocupará, o al menos eso se espera, de dar respuesta a los informes de *bugs* que lleguen a través del *Bug Tracking System* o BTS (véase el cuadro 2.3, *Bug Tracking System*), y de mantener actualizado el paquete subiendo una nueva versión cuando se actualice la versión *upstream*.

4.6 Debian como distribución.

Recordemos que Debian es una distribución de *software* libre. Es decir, como vimos en el cuadro 1.7, *Distribuciones GNU/Linux*, un conjunto de programas y paquetes. Éste es el resultado de las cadenas de prácticas en las que se articula Debian. Lo que ha distinguido a Debian desde su origen es la calidad de sus herramientas de gestión de paquetes, que permiten la integración de *software* muy heterogéneo en el mismo sistema, resolviendo los problemas de dependencias en-

tre unos paquetes y otros. Como se dice en la *wiki* del Proyecto: «The package management system binds all the software on a Debian system together» (Wiki.debian.org, s.f.[w]). Estas herramientas han sido una de las razones del enorme éxito de Debian como distribución.[20]

En el *post* «What is a Linux distribution?» el Desarrollador de Debian Lars Wirzenius (2012) explica cómo entiende qué es una distribución:

> [...] Here's my take: a Linux distribution is defined by the following things:
>
> - It's a specific, chosen set of upstream projects. The selection criteria vary wildly between distros, and may be based on things like purpose, quality, licenses, etc. [...]
> - It's a set of changes made to the upstream code, and specific configurations of the upstream code, in order make all the software work well together. [...]
> - It's the tools, processes, policies, and workflows used to develop the distribution.
> - It's the tools provided to the people using the distribution to install and manage their systems.
> - It's the people and companies who develop the distribution and support its users, and their shared values and purpose. This may be a real community, or just the kind of community that exists as a web page. [...]
>
> The most important differentiating factors between the various distros are not the tools, but the community values, the perceived purpose of the distro, and the resultant criteria for choosing what to include and the workflow for developing the distro. [...]

[20] Sobre estas herramientas y su interrelación véase el cuadro 4.1, *Algunos conceptos y términos sobre paquetes*, y las páginas Wiki.debian.org (s.f.[w]) y Debian.org/doc (s.f.[g]).

Esta definición pone de manifiesto varios elementos muy importantes. En primer lugar, la distribución establece relaciones entre los desarrolladores originales y los usuarios. No sólo entre cada uno de ellos y Debian, sino entre ellos entre sí, entre los usuarios y los proyectos *upstream*. Se establece entonces como un punto de paso entre ambos, delegando determinados procesos (como la instalación de nuevo *software*) a ciertos dispositivos técnicos (como los paquetes), aunque de un modo que bien puede ser contestado por los proyectos *upstream*, como veremos en la próxima sección. En esta labor de mediación es fundamental, y eso distingue a las distribuciones de otras maneras de acceder al *software*, el papel de los seres humanos. Como escribe un Desarrollador de Debian, el mantenedor se convierte en un filtro humano:

> [...] all Debian packages **must** be human-generated, human-reviewed, human-submitted. This means, a person must think each packaged piece of code is worth packaging, is stable enough and provides value to users as it is, and is fit for being part of a stable release (Wolf, 2011a).

Además, ésta es una vinculación que los transforma. La distribución es un objeto que reúne, como se ve aquí, diferentes sujetos, herramientas, comunidades, valores, objetivos, etc. Es decir, es un objeto híbrido y un mediador (véanse las secciones 4.2.2, *Mediadores e intermediarios. Las figuras del* bug *y del* hacker, y 4.4, *El código como objeto híbrido*). Que Debian tenga éxito, que sea usada y para qué, depende de cómo sea capaz de vincularse con otros proyectos, sujetos y lugares. Como distribución es un mediador también en la medida en que pretende producir cambios en los objetivos y el comportamiento de los agentes con los que se vincula. Por ejemplo, intentando que un proyecto *upstream* cambie algún elemento o su licencia, o contribuyendo a que los usuarios usen preferentemente *software* y *firmware* libres, situando los elementos no libres en otra parte del archivo (véase la sección 5.2.3, *Las licencias en Debian*).

4.6.1 Debian como *Proxy*. *Upstream* y *downstream*.

Al poner a disposición de los usuarios una colección de *software* en su mayor parte creada por otros individuos o grupos, Debian puede ser considerado como un *proxy* en la circulación de *software* libre:

> a compilation of the size of Debian can be considered a good proxy of libre software in general, thus offering a macroscopic view of the libre software landscape (Gonzalez-Barahona *et al.* 2009, pág. 266).

Consideremos un momento el uso de la palabra *proxy*. Algunos de los sentidos que ofrece *The New Oxford Dictionary of English* son los siguientes: «The authority to represent someone else, especially in voting or marrying»; «A person authorized to act on behalf of another»; «A figure that can be used to represent the value of something in a calculation» (Pearsall, 1998, pág. 1494). Dado el contexto del artículo, el sentido más cercano es el tercero ofrecido, puesto que efectivamente es razonable esperar que el estudio de Debian ofrezca conclusiones válidas respecto al *software* libre en general. Pero la palabra *proxy* tiene también un uso muy concreto en informática. Así define *Wikipedia* un *servidor proxy* (Wikipedia, s.f.[f]):

> In computer networks, a proxy server is a server (a computer system or an application) that acts as an intermediary for requests from clients seeking resources from other servers. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource available from a different server and the proxy server evaluates the request as a way to simplify and control its complexity. Proxies were invented to add structure and encapsulation to distributed systems. Today, most proxies are web proxies, facilitating access to content on the World Wide Web and providing anonymity.

Y el caso es que la distribución Debian actúa efectivamente (al menos metafóricamente) como un *proxy* en este sentido: cuando un usuario necesita un

determinado *software*, puede conseguirlo a partir de su desarrollador original, o alternativamente de una distribución como Debian, que lo empaqueta y distribuye (adaptándolo y modificándolo cuando es necesario) a partir de la versión *upstream*. Del mismo modo, muchos desarrolladores *upstream* entienden que una manera importante de distribuir su *software* es, bien empaquetarlo ellos mismos en el formato usado en las distribuciones más importantes (`.deb` en el caso de Debian), bien prepararlo para que algún Desarrollador Debian pueda empaquetarlo con facilidad e integrarlo en el archivo de la distribución. Éste es en gran medida el papel de una distribución como Debian en la circulación y la distribución del *software* libre.

Proxy es también, según el diccionario citado, alguien autorizado para actuar o representar a otro. Y también en buena medida es eso Debian: un actor que representa a los creadores de *software* libre frente a los usuarios, también a éstos frente a aquéllos (por ejemplo, enviándoles los informes de fallo o *bugs*). Sin olvidar que es también un punto de paso necesario entre aquellos desarrolladores y las distribuciones derivadas como Ubuntu. En todos estos casos, alterando o modificando las intenciones y objetivos de los demás actores, del mismo modo que éstos alteran y modifican los de Debian: en definitiva, convirtiéndose en un actor-red y contribuyendo a constituir la red del *software* libre. Cuando hablemos de la obligación de recibir (véase la sección 5.3, *La obligación de recibir*) plantearemos algunas cuestiones a este respecto. Como «parche», «agente» y otros términos, empezando por «red», «proxy» es un término surgido (en alguno de los sentidos considerados) de las prácticas y objetos producidos en el campo de la tecnología, especialmente informática, cuyo poder descriptivo trasciende el campo empírico en el que surge.

Upstream se refiere a los desarrolladores originales del *software* que Debian empaqueta en su distribución, como el *kernel* Linux, GNOME, el lenguaje de programación `python`, el servidor *Apache*, etc. *Downstream* se refiere fundamentalmente a las distribuciones derivadas que se basan en Debian, realizando alguna modificación sobre ésta, como Ubuntu o Arch Linux, o que usan *software* desarrollado en Debian. Los tamaños e importancia relativos de los lugares y proyectos que funcionan como *upstream* y *downstream* son muy variables. Los que se han mencionado en este párrafo son muy relevantes, y suponen la coor-

dinación de muchos individuos. Pero también hay proyectos realizados por una sola persona, o con un nicho de usuarios muy reducido. *Downstream* puede ser una pequeña distribución especializada o un proyecto comercial tan conocido y relevante como Ubuntu.

Ésta es una metáfora sobre las relaciones espaciales en el colectivo del *software* libre que pone el acento en los flujos, situando algunos puntos, en este caso Debian, como punto de paso obligatorio. Estos términos reflejan una relación direccional de origen y autoría, pero no de la contribución o del don (véase el capítulo 5, *El don del* software *libre*). Las contribuciones o donaciones se mueven también *upstream*, a través del envío de parches o informes de error, por ejemplo. Por ejemplo, es muy importante la contribución de los miembros del *Debian Kernel Team* al desarrollo del *kernel* Linux,[21] el núcleo del cualquier sistema GNU/Linux, y que es utilizado en todas las distribuciones. Es cierto que muchos proyectos *upstream* donan su *software*, pero también lo es que así pueden usar la infraestructura del Proyecto Debian para distribuir su *software*, para que sea testado, y se reporten informes de fallos.

Algunas de las ventajas para los desarrolladores *upstream* se detallan en la página *wiki* (Wiki.debian.org, s.f.[x]) del Proyecto. Entre éstas se mencionan la base de usuarios que pueden instalar fácilmente el programa, el que lo puedan hacer de manera segura, su integración con el resto del sistema, o la adaptación del programa a otras arquitecturas. También se ofrecen (Wiki.debian.org, s.f.[y]) una serie de consejos para que un desarrollador o proyecto faciliten que su código pueda integrarse en Debian. Estrictamente sólo sería necesario que la licencia fuese compatible con las *Debian Free Software Guidelines* (véase la sección 5.2.3, *Las licencias en Debian*), pero hay otras cosas que el desarrollador original puede hacer para facilitar la labor del Desarrollador o Mantenedor de Debian. En resumen, lo más importante sería ofrecer versiones del código fuente que no dependan de código externo ni de la inclusión de bibliotecas de *software* con versiones diferentes a las ya incluidas en Debian. Así se podría construir el binario sólo a partir de los paquetes fuente que ya están en el archivo de Debian. También se ofrecen recomendaciones sobre cómo preparar la documentación necesaria. En definitiva, se trata de desarrollar el *software* teniendo en cuenta los requisitos de

[21] Sobre esto véase Zacchiroli (2012a).

su integración en un sistema más amplio.

Un desarrollador *upstream* puede seguir o no estos consejos, por lo que la relación de Debian con unos y otros no siempre es fácil. Un miembro del *Debian Perl Group*[22] me contaba así la relación con el *upstream* de desarrolladores de Perl:

> **tincho:** He trabajado mucho en el *Debian Perl Group* como empaquetador, porque bueno, es un grupo genial para trabajar. Y además lo de empaquetar ahí es bastante sencillo. [...] no hay que volverse tan loco con las librerías[23] y con *upstream*, el *upstream* de Perl es muy bueno. Es un trabajo lindo, agradable. Con gente de CPAN,[24] de Perl, de *upstream*, había una muy buena relación [...] el *upstream* es muy bueno en general. Nos escuchan mucho, nos piden opinión. Perl es una cosa muy amorfa, no existe como una comunidad como Debian, pero digamos que todo se centra alrededor de CPAN. Y hay gente muy activa en CPAN, gente que está haciendo trabajos de QA (*Quality Assurance*) en CPAN. Esta gente ha venido a nosotros diciéndonos, bueno, estamos pensando hacer unos lineamientos y ustedes son nuestro mejor *downstream*. Y bueno, hablemos a ver cómo podemos hacer mejor para todos. Sí, muy bien. Y cuando hemos propuesto cosas las han mejorado, o sea que es el mejor *upstream* posible.

Vemos que la buena relación supone un interés mutuo, una relación beneficiosa para ambas partes. Esta relación se ve muy facilitada por el hecho de compartir determinadas formas de práctica y determinadas convenciones, por compartir determinada cultura:[25]

[22] Perl es un lenguaje de programación muy utilizado. Sobre el *Debian Perl Group* véase el final de la sección 3.2.1, *Los Desarrolladores Individuales: doocracia*, y Wiki.debian.org (s.f.[k]).

[23] La traducción que venimos usando es biblioteca. Es corriente tanto el uso de biblioteca como el de librería para traducir *library*.

[24] CPAN es el *Comprehensive Perl Archive Network* (Cpan.org, s.f.), un repositorio de módulos de Perl, que extienden su funcionalidad.

[25] Otro ejemplo de buena relación con *upstream* es la colaboración con el proyecto GNU (Gnu.org, s.f.[a]). Nos limitamos aquí a remitir a un mensaje en el que se resumen algunos aspectos de esta relación, y a partir del que se puede profundizar en la cuestión, en Debian-project (2011a). Véase también Zacchiroli (2011b).

> **tincho:** La comunidad usa estándares que tienen mucho tiempo, muy aceptados, muy razonables. Es una comunidad particular porque es solamente los que hacen librerías, no es *Perl Core*.[26] Es otra cosa que la que tenés en Java, en Python, Ruby, [...] La diferencia con los *upstream* de todos esos es infernal, porque todas las demás son comunidades más modernas, que no llevan la experiencia de Perl. Existe hace como 30 años, 25 creo o por ahí. Es gente que viene más de una escuela antigua. Esa gente está antes de Linux haciendo Perl. Más antigua escuela, como Debian también, porque Debian está acostumbrado al `tar.gz`, `configure`, `make`, `make install`. Bueno, Perl es así. Básicamente, a la vieja escuela.

No todas las relaciones con *upstream* funcionan así. Recordemos que, en relación al problema relativo a Python analizado en 2.1, *Bug #573745*, uno de los problemas era la diferente manera de instalar y actualizar los paquetes que hay en Debian y en el *upstream* de Python. Para algunos de los autores originales (sean individuos, proyectos o comunidades), una distribución como Debian es en realidad un obstáculo que distorsiona la relación entre ellos y los usuarios, dificultando de hecho que éstos puedan usar la última versión del *software* del modo en que los desarrolladores *upstream* piensan que deberían hacerlo. En la misma entrevista:

> **tincho:** [...] otros equipos donde tienen que pelearse con un *upstream* que a veces es hasta hostil, porque piensa que las distribuciones son un estorbo entre el desarrollador y el usuario. Hay una gran diferencia.
>
> **Fernando:** ¿Qué *upstreams* funcionan así?
>
> **tincho:** Java, Ruby, Python, bastantes. No he estado involucrado, pero ya lo ves solamente en la manera en la que hacen las cosas y en los sitios *web* donde ellos tratan de poner sus paquetes para las distribuciones, en vez de hablar con las distribuciones para que las distribuciones se encarguen. Nunca he estado involucrado, pero he escuchado gente de Ruby [en Debian], que ha desistido de empaquetar, porque el *upstream* es terrible, realmente no quiere las distribuciones. Ellos

[26] *Perl Core* se refiere a las librerías estándar que acompañan al intérprete de Perl, `perl`.

> piensan que el usuario tiene que bajarse las cosas directamente, que es lo que un poco intentan hacer Chrome, Firefox, con sus actualizaciones automáticas y todo eso.

Veamos uno de estos casos como ejemplo, el de Ruby. Me limitaré a mencionar como estaba la situación en 2010. Un Desarrollador de Debian y miembro del equipo que mantenía Ruby en Debian escribía las siguientes conclusiones en su *blog*, después de repasar algunos de los problemas y conflictos en la relación entre Debian y Ruby (Nussbaum, 2010b):[27]

> The Ruby developers (or expert users) community is generally very hostile towards Debian. Many harsh words and insults are used in discussions mentioning Debian in the ruby-core@ mailing list, and there are frequent recommendations to avoid the Debian packages and install from sources, which is quite demotivating (Actually, what prompted that blog post is someone calling Debian-specific changes unforgivable).
>
> This atmosphere makes it hard to recruit people, and the Debian Ruby teams are completely understaffed, which is clearly the major blocker to improving the situation further. We are 3 co-maintainers for the interpreter packages, and I'm the only Debian Developer that is really active on a regular basis in the pkg-ruby-extras team (that does libraries and applications packaging). We desperately need help, but at the same time I have no time to improve our documentation and make it easier to join the team.

Los problemas que menciona en ese *post* implican la diferencia entre la cultura de Ruby de usar siempre versiones muy modernas de su *software*, incluso si impiden la compatibilidad con desarrollos anteriores, frente a la cultura de Debian de primar la estabilidad y la solidez. Y la portabilidad, con Ruby soportado oficialmente sólo en una arquitectura, en lugar de la decena que debe soportar Debian. Todo esto hace que los desarrolladores de Ruby recomienden no usar las

[27] Véanse también Nussbaum (2010a, 2011).

herramientas de Debian para instalar su entorno de desarrollo. Quizá el problema principal sea la diferente concepción sobre el uso de bibliotecas (véase la página 64) por parte de diferentes programas (Wolf, 2010). Frente a Ruby (y otros entornos similares), Debian insiste en instalar una vez cada biblioteca, de tal modo que sea usada por todos los programas que la necesiten. Por ello se han de empaquetar de manera separada de los programas que las usan. Para que esta situación sea mantenible y no se complique en exceso es necesario limitar todo lo posible el número de versiones diferentes de una misma biblioteca que se instalan. Pero mientras los desarrolladores de aplicaciones en Ruby piensan fundamentalmente en las necesidades de otros Desarrolladores (que necesitan disponer de las últimas versiones de cada biblioteca, no de una versión estable), los de Debian piensan fundamentalmente en las de los usuarios y administradores de sistemas, que necesitan facilidad de instalación, de uso, y estabilidad (Wolf, 2010, 2011a).[28]

En el fondo, en éste y otros casos parecidos,[29] lo que hay aquí es una lucha por definir el modo en que los usuarios usan el *software*, por convertirse en el punto de paso obligatorio entre los usuarios y el *software* que se usa. Hay implicadas diferentes concepciones de la seguridad o la estabilidad, el ritmo de actualizaciones, la redundancia y la interacción con otras partes del sistema, o el uso de diferentes esquemas de licenciamiento. Pero todo esto no son sino diferentes concepciones de cuáles deben ser los fines de los usuarios, un intento de traducirlos de diferentes maneras (en el caso de Debian, a través de su «Contrato Social»), de cuál es la manera adecuada de relacionarse con el *software*. La consecuencia de esta disputa es la modificación de los actores, y la redistribución de la agencia a lo largo de toda esta red de vinculaciones.

En los casos más extremos se puede hablar incluso de *upstream* hostil. A consecuencia de casos como estos, se añadió el siguiente párrafo a la «Debian Developer's Reference» (Debian.org/doc, s.f.[e], cap. 3):

If you find that the upstream developers are or become hostile towards

[28] Puede encontrarse una discusión más actual de estas cuestiones (en absoluto centrada en Ruby) y el papel de Debian como distribución en Wirzenius (2018).

[29] Hay muchos. Mencionaré sólo uno que acabó con la petición de *upstream* de no ser empaquetado para Debian: ownCloud. Puede verse un mensaje sobre esta cuestión en Alioth-lists.debian.net (2016).

> Debian or the free software community, you may want to re-consider the need to include the software in Debian. Sometimes the social cost to the Debian community is not worth the benefits the software may bring.

La relación con *downstream* es también compleja y no siempre fácil. Nos limitaremos a comentar muy brevemente la relación de Debian con la distribución derivada más usada y conocida, Ubuntu (Ubuntu.com, s.f.[a]), desarrollada por la compañía Canonical. Desarrollar propiamente esta cuestión daría para otra tesis doctoral, así que nos limitaremos a unos comentarios muy generales. Ubuntu surgió en 2004 como una derivación (un *fork*) de Debian, y se ha convertido a su vez en el *upstream* de otras distribuciones. Una buena parte de los desarrolladores de Ubuntu, sobre todo al principio, son al mismo tiempo Desarrolladores de Debian.

El hecho es que la relación entre ambas distribuciones ha sido una fuente más o menos constante de tensiones. Ya vimos como una de las cuestiones surgidas en la controversia sobre `python` analizada era la interpretación, por parte de quienes escribieron el informe de fallo, de que su mantenedor estaba más interesado en mantener propiamente el paquete para Ubuntu que para Debian. También estaba presente la relación con Ubuntu en la controversia sobre la modificación del proceso de publicación de la distribución, mencionada en la sección 3.2.2, *Consenso*. Ya vimos (en la sección 3.2.4, *Concentración de poder y los límites internos de la doocracia*) también que causó preocupación el hecho de que Canonical contratara a unos cuantos Desarrolladores de Debian para desarrollar Ubuntu, lo que provocó que bastantes de ellos disminuyeran su contribución a Debian. Por último, también ha habido numerosas controversias en torno a las diferentes actitudes respecto a la comunidad del *software* libre y los usuarios, pero sobre todo respecto a la reciprocidad, a la contribución de los desarrollos realizados en Ubuntu de vuelta a Debian.

Pero en general, estas tensiones se han ido rebajando con el tiempo. En mayo de 2010, el DPL escribía un mensaje en la lista de correo `debian-project`[30] sobre

[30] En Debian-project (2010c). En Debian-project (2010d) hay un resumen de las opiniones de algunos Desarrolladores de Debian sobre los éxitos y fracasos de la relación con Ubuntu. La presentación usada se puede ver en Zacchiroli (2010a). Puede verse una continuación el siguiente año en

una reunión con miembros de Ubuntu en el *Ubuntu Developer Summit*, sobre las posibilidades de mejorar la colaboración. Esta colaboración sería beneficiosa para ambas partes. Para Ubuntu, porque es de donde obtiene casi todos los paquetes que distribuye. Para Debian, porque aumenta su base de usuarios, aunque sea de manera indirecta, y puede beneficiarse de los informes de fallo que provengan de Ubuntu. Una cuestión interesante de esa reunión es que muestra como Ubuntu puede ser a la vez *downstream* y *upstream* de Debian, en tanto que no sólo se construye a partir de ella sino que es el origen del *software* desarrollado en Ubuntu.

4.7 Sobre algunas dimensiones de la práctica de empaquetado.

Hemos visto como la práctica básica en Debian, la creación y mantenimiento de paquetes, crea o contribuye a crear toda una serie de relaciones y vínculos que dan consistencia al colectivo, tanto si hablamos del colectivo de Debian en concreto como del colectivo del *software* libre en general. Que esta actividad sea una práctica de vinculación permite también arrojar algo de luz, disolviéndolas, sobre lo que aparece a primera vista como una serie de oposiciones dualistas, algunas de las cuales mencionábamos al inicio del capítulo.

En primer lugar, la labor del mantenedor de Debian es al mismo tiempo individual y social. Es cierto que cada vez más se tiende a la formación de equipos para mantener paquetes relacionados, pero también lo es que la gran mayoría están asociados al nombre individual de su mantenedor. Y que la práctica material de empaquetado se realiza individualmente frente al ordenador. Si además se constituye como una práctica social no es sólo porque, como toda práctica, está condicionada por las relaciones con otros sujetos y las convenciones asociadas, sino porque es esta práctica la que crea el colectivo sociotécnico. Al seguir, por ejemplo, las buenas prácticas descritas en la *Developer's Reference* y el *Debian Policy Manual*, no se aplican sin más normas técnicas de eficiencia, sino que se da consistencia a esas convenciones y se articulan las relaciones entre diferentes agentes.

Debian-project (2011b).

Es más, precisamente porque es una práctica de vinculación podemos ver que desaparece la oposición entre la acción de un individuo y el marco social que la condiciona. El «marco social» relevante no es otro que esos vínculos que emergen de la cadena de prácticas (para una crítica de la idea de contexto social, véanse Latour (1996, págs. 137-138) y Latour (2005, págs. 165-173)). Ya hemos considerado previamente, en la sección 2.2.3, *La producción práctica de la distinción*, la construcción de la oposición entre lo técnico y lo social. En parte, esta oposición es una reformulación de la anterior. Si esta práctica es individual, es porque es técnica. Pero la elaboración de un paquete no depende sólo de criterios de eficiencia técnica, de que el *software* funcione, sino que responda también a criterios convencionales y valorativos, que los miembros de Debian llamarían sociales.

De manera similar, hay que insistir en la idea de que esta práctica técnica es también, siempre, una práctica legal. Lo que no puede faltar nunca en un paquete fuente que se suba a Debian es una licencia compatible con las *DFSG*, lo que suele implicar que lleve también el código fuente. De hecho, subir un paquete por primera vez al archivo de Debian supone una revisión exhaustiva de su licencia por parte del equipo FTP Team. Sobre esta cuestión se profundizará en la sección 5.2.3, *Las licencias en Debian*, pero se puede afirmar que no es posible mantener paquetes en Debian sin un conocimiento más o menos profundo de las cuestiones legales que afectan al *software* y la propiedad intelectual. Esto se puede generalizar a desarrolladores de *software* libre y *hackers* en general:

> Many hackers, understood to be technologists, became legal thinkers and tinkerers, undergoing legal training in the context of the F/OSS project while building a corpus of liberal legal theory that links software to speech and freedom.
>
> [...]
>
> Hackers have been in part successful in this political fight because of their facility with the law; because of years of intensive technical training, they have not only easily adopted the law but also tinkered with it to suit their needs (Coleman, 2013, pág. 183).

Lo veremos en la sección 5.2.1, *Las licencias libres*. Y como veremos tam-

bién más adelante (sección 5.2.2, *Sobre el sentido de «libre» en el* software *libre y la cultura libre*), esa dimensión legal está a su vez ligada a una determinada concepción de valores éticos, sociales y políticos, que determinan qué *software* se empaqueta y cómo, más allá de su mera funcionalidad instrumental, expresando así determinados valores.

Por último, podemos preguntarnos por qué se empaqueta voluntariamente para Debian, sin en principio recibir nada a cambio, si se puede hacer para uno sólo. Esto nos lleva a preguntarnos sobre el don del *software*, tema del próximo capítulo. Pero antes, y para ello, nos detendremos brevemente a considerar un aspecto más de la creación y uso de dispositivos tecnológicos en Debian.

4.8 Repositorios de *software* libre.

Si de algún dispositivo tecnológico se puede decir con exactitud lo que Latour decía (véase la sección 4.4, *El código como objeto híbrido*) de los cuasi-objetos o de las redes, es de los repositorios: que «son a la vez reales como la naturaleza, narradas como el discurso, colectivas como la sociedad» (Latour, 2007, pág. 22).

Y es que para entender las características del código y del *software*, un elemento fundamental es la comprensión del lugar donde éste se desarrolla. Cada vez más, este lugar consiste en repositorios digitales gestionados por sistemas de control de versiones (véase el cuadro 2.5, *Sistemas de Control de Versiones y Repositorios*), y asociados normalmente a sistemas de seguimiento de fallos o espacios donde alojar la documentación. Además de ayudar a la comprensión del proceso de desarrollo de código, considerarlos es importante porque, como afirman De Paoli y D'Andrea (2008, pág. 1), «technologies may embody organizational rules, hence becoming the core of coordination efforts in such communities». Sobre el papel de los dispositivos en la organización de las comunidades formadas en torno al *software* libre, y especialmente de los sistemas de control de versiones, son también interesantes Lanzara y Morner (2005), y Shaik y Cornford (2004).

Así, los repositorios son «reales, como la naturaleza», en tanto que son una condición material para el desarrollo y distribución del *software*. Los más visibles son los repositorios para la *distribución* de *software*, desde los que es posible

descargar e instalar programas informáticos. Algunos de los más conocidos en el ámbito del *software* libre son por ejemplo CPAN para el lenguaje de programación Perl (Cpan.org, s.f.), CTAN para TEX y LATEX (Ctan.org, s.f.), PyPi para Python (Pypi.org, s.f.) o CRAN para R (Cran.r-project.org, s.f.). Y por supuesto, los repositorios ofrecidos por cualquier distribución GNU/Linux, en nuestro caso el archivo de Debian (Debian.org, s.f.[q]). Estos repositorios almacenan recursos de manera integrada y permiten instalar, actualizar y configurar diferentes programas de manera sencilla, coherente y unificada.

Pero de hecho, para que estos repositorios puedan existir es necesaria también la existencia de repositorios para el *desarrollo* de *software*. Es decir, repositorios en los que se desarrolla el *software* que luego estará disponible para su distribución. Al distinguir repositorios para la distribución y repositorios para el desarrollo estamos haciendo referencia a dos funciones, no necesariamente a dos mecanismos o localizaciones diferentes. Un ejemplo en el que están separados es cualquier distribución GNU/Linux: por un lado tenemos un archivo de la distribución, y por el otro al menos un conjunto de repositorios *upstream* de desarrollo individuales para cada programa o proyecto que se integra en la distribución, sin contar con los posibles repositorios intermedios que pueda usa un Desarrollador o un equipo de Debian para realizar el trabajo de empaquetado.

En cualquier caso, uno de los rasgos más importantes que diferencia ambos tipos de repositorio suele ser el uso de un sistema de control de versiones, que permiten la gestión del desarrollo de *software*, guardando copias de las distintas versiones del mismo. Su principal ventaja es que permite la colaboración simultánea por parte de diferentes individuos en una misma pieza de *software*, sin que sus contribuciones entren en conflicto. También el hecho de que almacenan metadatos sobre el código y cada modificación del mismo, lo que permite hacer búsquedas complejas en ellos. Los primeros sistemas eran centralizados (los más conocidos eran `cvs` y `subversion`), pero desde hace tiempo los más usados son los sistemas de control de versiones distribuidos, caracterizados porque no existe un servidor central que actúe como único repositorio autorizado, sino que cada copia local es un repositorio completo y funcional.

El sistema más usado, y que se ha ido imponiendo sobre los demás, es `git`,

desarrollado por el creador de Linux Linus Torvalds. Un flujo de trabajo muy básico empezaría clonando (copiando) un repositorio remoto o *upstream* en el propio ordenador. Esta clonación no copia sólo la última versión de los contenidos, sino toda la historia de su desarrollo. Ésta será una copia completa del código fuente, y pueden clonarse otras copias a partir de ella, sin necesidad de recurrir a ningún repositorio central. A continuación se pueden editar localmente los archivos que se quiera. Cuando el desarrollador lo desee puede agregar esos cambios a la historia de versiones de `git`, haciendo lo que se llama un *commit* y creando una versión nueva del código fuente, que añade los cambios realizados. Cada *commit* incluye el nombre de su autor y un comentario de los cambios realizados. Lo que se guarda son solamente los cambios respecto a la versión anterior. A partir de estos cambios es posible reconstruir una versión anterior del código, así como examinar el historial de cambios. Si se tiene permiso, se pueden subir estos cambios a otro repositorio, por ejemplo a aquél desde el que se clonó y que estará administrado por el proyecto *upstream*, o publicar el repositorio independientemente. Así se pueden distribuir fácilmente los cambios que se realizan sobre el código, sin necesidad de mandar manualmente parches con los cambios o versiones completas del código.

En Debian cada Desarrollador o equipo puede elegir aquellas herramientas con las que trabaja. Durante mucho tiempo, desde 2003,[31] los repositorios (entre otras herramientas y servicios) de Debian se alojaban en el servidor *alioth*, donde se podían usar varios sistemas distintos de control de versiones. Pero debido a algunos problemas con el *software* (GForge) que hacía funcionar este servidor, entre los años 2017 y 2018 se produjo la transición de muchos de esos servicios y herramientas desde *alioth* hasta *salsa* (Salsa.debian.org, s.f.[a]), basado en el *software* GitLab y que sólo soporta `git` como sistema de control de versiones. Aunque no podemos dar aquí ni siquiera un resumen, esta transición implicó elementos muy interesantes para pensar la cuestión de una infraestructura propia y libre, sus condicionamientos y requerimientos.[32]

En cualquier caso, también en Debian se está imponiendo el uso de `git` fren-

[31] Véase Debian-devel-announce (2003).
[32] Se puede llegar a muchas de las discusiones relevantes a partir de Wiki.debian.org (s.f.[z]) y de los enlaces que allí aparecen.

te a otras formas de control del código fuente. Siempre ha sido muy fácil descargar el código fuente de un paquete determinado desde el archivo de Debian, pero con el uso de los sistemas de control de versiones lo que se facilita es la descarga del código todavía en desarrollo. Para esto, el archivo `debian/control` (véase la sección 4.5, *El empaquetado como práctica sociotécnica*) debe indicar dónde está el repositorio que contiene el código (Wiki.debian.org, s.f.[aa]). Hay varios modos de usar `git` para el desarrollo de paquetes (Wiki.debian.org, s.f.[ab]), pero la cuestión principal es que esto no sólo puede facilitar el flujo de trabajo de un Desarrollador o Mantenedor, sino facilitar la colaboración tanto con *upstream* como con *downstream* y los usuarios, en tanto que facilita el envío de modificaciones al código fuente. Estos dispositivos son también una forma de incluir a los usuarios en los flujos de colaboración y participación.

En 2014 se empezó a trabajar en una *Debian Enhancement Proposal* (véase 3.3, Debian Enhancement Proposals Workflow) para estandarizar la estructura de los repositorios `git` usados en el desarrollo de paquetes, y las convenciones sobre su uso, con el objetivo precisamente de facilitar la colaboración (Dep-team, s.f.[d]). Así argumentaba el Desarrollador que proponía esta DEP sus ventajas en un correo a la lista `debian-devel` (Debian-devel, 2014d):

> To: debian-devel@lists.debian.org
> Subject: Standardizing the layout of git packaging repositories
> From: Raphael Hertzog <hertzog@debian.org>
> Date: Fri, 15 Aug 2014 16:16:01 +0200
>
> Hello,
> while git is the most popular VCS for packaging, there's no clear rules
> on what you can expect in a random git packaging repository listed in Vcs-Git.
> I would like to fix this so that:
>
> - we can extract more useful data from the git repositories
> - we can more easily share our git repositories with upstreams and downstreams
> - we start to converge on some conventions even though we might continue to use different git helper tools
>
> I want to use the DEP process for this. […]

En ese correo vemos entonces tres elementos importantes para una consideración de los dispositivos técnicos en general y de los sistemas de control de

versiones en particular: el uso de los metadatos, la facilidad para compartir y las convenciones implicadas en el desarrollo de *software*. Por otra parte, como vimos en la sección 3.2.5, *Doocracia y agencia: la mediación técnica*, la posibilidad y el permiso para hacer *commit* en los repositorios del Proyecto es una de las maneras en que se expresan las posiciones de poder y responsabilidad. Es una de las medidas de la capacidad de agencia de cada individuo, y una de las situaciones donde se expresa el funcionamiento de la doocracia.

Una de las funcionalidades más apreciadas de `git` es su facilidad para crear y gestionar diferentes ramas de desarrollo, líneas independientes de cambios. Se puede trabajar en una rama experimental, o para desarrollar una característica nueva, o cualquier conjunto de cambios relacionados, independientemente de la línea de desarrollo principal. Esto facilita enormemente la colaboración entre diferentes personas e incluso proyectos. Es posible mantener un repositorio con ramas para colaborar con diferentes proyectos o individuos, siendo muy fácil la integración de cambios de una rama a otra. Todo esto facilita la apertura de los repositorios y la colaboración de usuarios externos.

Más allá de este flujo de trabajo básico se pueden producir otros mucho más complejos, como la posibilidad de añadir diferentes repositorios remotos, elegir individualmente los *commits* de otra rama u otro desarrollador que se van a integrar, compilaciones automáticas del código cuando éste se cambia, avisos automáticos de los cambios, interacción automatizada con otros repositorios, incluso aunque usen otros sistemas, por citar algunas posibilidades. Todo esto es posible realizarlo manualmente, sin estos sistemas, pero la complejidad de las tareas requeridas aumentaría enormemente.

Un Desarrollador de Debian exponía así las ventajas de estos sistemas:

> **dato:** Las ventajas de los sistemas distribuidos son dos. Primera, una especie de democratización de las herramientas. Cuando tienes un sistema centralizado, está en un servidor, y cualquier persona que quiere hacer cambios tiene que tener acceso de escritura a ese servidor. Pero tú no le puedes dar acceso de escritura a cualquier persona, tienes que

> poner una política para cuando das el acceso y cuando no. Entonces se crean dos clases de desarrolladores o contribuidores, los que tienen acceso y los que no. El que está empezando tiene una cantidad limitada de herramientas, no puede hacer *commits*, no puede crear un *branch*, tiene que funcionar a base de parches que luego envía, etc etc. Está como en una segunda clase. En un sistema distribuido cualquier persona puede hacer una copia del repositorio y trabajar ahí al mismo nivel, con el mismo *set* de herramientas que los desarrolladores con acceso están teniendo. Lo único es que no va a poder publicar en el repositorio principal, pero puede publicar en un servidor suyo o en cualquiera de los muchos sitios de publicación gratuita que hay. Y lo interesante es que en esa copia puede hacer todo lo que pueden hacer los desarrolladores con acceso de escritura. Lo único que no puedes es hacer *push* a www.proyecto.org pero puedes hacerlo a miservidor.com. Eso me parece a mí la primera, es un cambio realmente de cómo trabajamos.
>
> Y otra cosa que me parece interesante es, por llamarlo así, por higiene personal, y por privacidad. Si tú tienes acceso de escritura a un sistema centralizado, y tienes una idea, pero que no está lista para ir a la principal, porque quieres experimentar, puedes crear una rama independiente. Todo el mundo la ve, y si era una idea loca que luego te avergüenzas de haber tenido, pues todo el mundo lo ha visto y sabe que has intentado eso y que ha fracasado. En un sistema distribuido puedes hacer una copia y experimentar y no lo publicas, y si no llega a ninguna parte lo borras y nadie sabe nunca nada, y si llega a alguna parte, antes de publicarlo puedes arreglar las cosas, los mensajes de *commit* para que sea inteligible.

Algunas de las ventajas de este tipo de sistemas, y que son relevantes para la cuestión que estamos tratando, son pues, y además de las mencionadas, que permiten una atribución clara de la autoría distribuida;[33] permiten una diferenciación de permisos atribuidos a los participantes, pero incluso alguien sin permiso de escritura o modificación puede trabajar para mejorar el código; facilitan enor-

[33] Y resuelven así uno de los problemas que amenazan el crecimiento del procomún: el problema del reconocimiento de las aportaciones individuales como factor de motivación (Ortega y Rodríguez, 2011).

mente la experimentación; permiten el acceso tanto a las versiones publicadas como al producto en desarrollo; facilitan el desarrollo de versiones alternativas o *forks*; permiten la existencia de múltiples repositorios ligados entre sí; establecen protocolos claros de colaboración y de trabajo en común, con diferentes flujos de trabajo; suponen la apertura no sólo de los productos sino también de los procesos; y además dan la posibilidad de estudiar todo el proceso de desarrollo.[34]

Pero sobre todo, como decía otro Desarrollador:

> **vorlon:** When you are using a distributed version control system you have a lot more opportunities to experiment, trying new things out and sharing them in a structured fashion with other developers.

Lo importante es que permite compartir «in a structured fashion». Es decir, por todo lo que estamos viendo, además de reales son «colectivos, como la sociedad». Constituyen un sistema para manejar el flujo de aportaciones a los diferentes proyectos. Pueden cambiar el significado de la colaboración, en tanto que más que de aceptar obras terminadas, de lo que se trata es de poder aceptar incluso pequeñas modificaciones (*parches*) de múltiples autores al mismo producto final. Son sistemas que han surgido para resolver el problema de distribuir y proteger un determinado ámbito del procomún, además de permitir y facilitar la contribución a ese ámbito (véanse las secciones 5.2.4, *Procomún y cultura libre*, y 5.3, *La obligación de recibir*): «Use of open source and free software licenses and web-based hosting services repositories such as GitHub encourages people to collectively expand existing software tools, which often leads to their rapid evolution» (Manovich, 2013, pág. 337), todo ello sin la necesidad de la intervención de grandes empresas.[35] Como veremos más adelante, que la comunidad del *software* libre sea abierta requiere de algún modo que los repositorios en relación a los que se constituye sean abiertos, en diferentes grados y a través de diferentes procesos.

Que sean colectivos quiere decir entonces que establecen relaciones y vínculos entre actores por diferentes medios. Que algunos de estos sean tecnológicos es

[34] Como muestra especialmente el trabajo realizado por el grupo Libresoft (Libresoft.es, s.f.) y las herramientas informáticas que han desarrollado para ello.

[35] A pesar de la mención a GitHub (Github.com, s.f.), que sí lo es, puesto que hay alternativas.

lo que le da consistencia a esas relaciones, como venimos viendo. Kelty establece así su relación:

> Coordination is important because it collapses and resolves the distinction between technical and social forms into a meaningful whole for participants. On the one hand, there is the coordination and management of people; on the other, there is the coordination of source code, patches, fixes, bug reports, versions, and distributions –but together there is a meaningful technosocial practice of managing, decision-making, and accounting that leads to the collaborative production of complex software and networks (Kelty, 2008, págs. 210-211).
>
> [...]
>
> One such tool plays a very special role in the emergence of these organizations: Source Code Management systems (SCMs). SCMs are tools for coordinating people and code (Kelty, 2008, pág. 229).

Como veremos en el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*, es esta vinculación entre humanos y no humanos lo que caracteriza la constitución de un colectivo sociotécnico. Y así expresa esa relación Yuill (2008), añadiendo una característica muy interesante, relativa a la trazabilidad de los procesos:

> Code creation is an inherently social act. It involves processes of collaboration, consensus, and conflict resolution, and embodies social processes such as normalization and differentiation. Software development tools such as CVS implicitly formalize such proceses and, in doing so, potentially provide means of tracking them. As a result of this, forms of sociological analysis have developed based around «archaeological» studies of CVS repositories (Yuill, 2008, pág. 66).

Esta última característica nos lleva al último punto: los repositorios son también «narrados, como el discurso». Además del análisis sociológico, lo que facilita

el hecho de la trazabilidad de las colaboraciones es el análisis discursivo y narrativo (como vimos en las secciones 1.3, *Cuestiones de metodología*, y 2.1.5.2, *El repositorio `git` del Comité Técnico*). La existencia de los sistemas de control de versiones distribuidos cambia la naturaleza del código fuente, porque no sólo se distribuye el código, sino la historia de su construcción y desarrollo. Manovich (2001) afirma que la base de datos ha sustituido a la narrativa clásica como forma cultural. Pero hay bases de datos, como un `log` de `git`, un repositorio en un sistema de control de versiones distribuido o el mismo BTS, que cuentan historias. Al menos, sus usuarios pueden construir narrativas con su ayuda, historias que de hecho sólo se pueden construir con su ayuda. Los repositorios permiten así acceder a la memoria de las acciones sociales que los constituyen. Éstas están inscritas en él porque se guarda el registro de los mensajes de `commit`, de los cambios producidos en el código paso a paso. El código en un repositorio de este tipo es un objeto, pero un objeto que no es mudo, que lleva en sí las huellas de su constitución y que puede contar su propia historia.

La producción textual por parte de los miembros de Debian no se limita entonces a producir un registro enormemente rico de sus comunicaciones interpersonales por medios electrónicos, como las listas de correo o el IRC. También, y esto es quizás más significativo, produce un registro minucioso, tremendamente rico y abierto de las aportaciones técnicas en la infraestructura de colaboración y en el propio producto de su interacción, los programas distribuidos:

> Hackers build oral histories through their code commits, comments and recognition in common files included with source code, the Changelog, Authors and Contributors files, where and who contributed what to a piece of software, is noted. This everyday discourse and inscription develops a shared historical awareness about who contributed what, one that brings attention to the conditions of production or the nature of the contribution. Further, many of the technical tools that facilitate collaboration, such as CVS and Subversion [...] not only *enable collaboration* but provide *precise details of attribution*, that over time come to congeal into a richly documented palimpsest; though individual attribution is certainly accorded, these technological palimpsests,

> over time, reflect unmistakably that complicated pieces of software are held into place by a grand collaborative effort that far exceeds any one persons' contribution (Coleman, 2005, pág. 244).

En definitiva, los sistema que acabamos de considerar son dispositivos tecnológicos que contribuyen en gran medida a facilitar la circulación de lo que en el capítulo siguiente vamos a ver que constituye el don del *software*, y a constituir por lo tanto el procomún del *software* libre. Unos dispositivos que permiten una atención detallada y diferenciada a las prácticas concretas de colaboración y creación de ese procomún.

5 El don del *software* libre.

> *Más allá de la esfera de los intercambios, existen otros dominios, otra esfera constituida de todo aquello que los hombres imaginan que deben sustraer al intercambio, a la reciprocidad, a la rivalidad, de todo aquello que creen que deben conservar, preservar, incluso enriquecer.*
>
> —Godelier (1998, pág. 58).

Una vez considerados qué son, cómo se producen y cómo se distribuyen los paquetes en Debian, es el momento de considerarlos poniendo de relieve su aspecto de *software libre*, es decir como objetos que se donan. Ya hemos visto en la introducción qué es el *software* libre, y las cuatro libertades que conforman su definición (véase la página 23). Es ya un lugar común aplicar el concepto de «economías del don» (o al menos simplemente la de «don») para explicar la dinámica de las comunidades constituidas en torno al procomún, la llamada «economía colaborativa», y especialmente en torno al *software* libre (véanse por ejemplo Berdou, 2011; Bergquist y Ljungberg, 2001; Ghosh, 2005b; Kollock, 1999; Mansell y Berdou, 2010; Ortega y Rodríguez, 2011; Raymond, 2000; F. Turner, 2009; Zeitlyn, 2003). Lo que normalmente se intenta hacer con esta aplicación es ofrecer un modelo que dé cuenta de la forma alternativa que toman una serie de actividades, como el desarrollo de *software*, que en el mundo contemporáneo se sitúan preferentemente en el ámbito de los intercambios monetarios y de la actividad empresarial. Lo que se pretende poner de manifiesto entonces de esta actividad es su carácter de donación desinteresada (al menos aparentemente) del producto del

propio trabajo. Así, se asimilan a ese respecto lo que aparece como dos formas de vida situadas en extremos opuestos de un gradiente de modernidad cifrada en su relación con la tecnología: las formas de vida de *hackers* y desarrolladores de *software* libre por un lado, y las de aquellas sociedades tradicionales caracterizadas por una «economía del don» por parte de los antropólogos. La primera precaución al respecto ha de ser entonces resistir la tentación de exotizar a quienes se dedican a desarrollar *software* libre.

¿Qué se puede entender, en sentido general, como un don? En Graeber (2001) encontramos una definición suficientemente general que puede justificar su uso en este contexto:

> To give a gift is to transfer something without any immediate return, or guarantee that there will ever be one. This is the definition adopted by the MAUSS group (e.g., Godbout and Caillé 1998), and it seems about as good a general definition as we're going to get. It also makes it clear that the term can apply to an enormous variety of transactions, and that the term "gift economy" can apply to any not organized on market principles (Graeber, 2001, pág. 225).

¿Qué hay en el fenómeno del *software* libre que hace que sea tan recurrente el uso de la teoría antropológica sobre el don? En primer lugar, por supuesto, el hecho de que sean los propios desarrolladores de *software* quienes la usen para explicar su propia actividad, sobre todo a partir de la obra de Eric Raymond, desarrollador de *software* que llega a definirse como antropólogo aficionado y que introduce la caracterización del ámbito del *software* libre como una economía del don.[1] Pero también el que sea una metáfora que aparentemente explica bien algunos de los elementos más llamativos del desarrollo del *software* libre: el hecho de que la mayor parte del trabajo sea voluntario, la colaboración desinteresada, o la falta de recompensa económica. Es decir, su oposición a la forma de mercancía en que el *software* propietario es producido.

[1] Esta concepción se desarrolla especialmente en Raymond (1999, 2000). Para una crítica del uso de esta caracterización por parte de Raymond, véase Kelty (2002, 2006).

En realidad, esta asimilación del desarrollo del *software* libre a las economías del don, y los dualismos que conlleva, puede ocultar tanto como ilumina, empezando por el mismo uso del término «economías del don» para referirse a las comunidades de creación colaborativa. Esto puede resultar engañoso por el uso del término «economías». Quizás uno de los elementos que más llame la atención del desarrollo del *software* libre sea su aparente irracionalidad económica, pero no hay que olvidar que para Mauss el don es un *hecho social total*, no sólo ni fundamentalmente económico. De hecho, poner esto de manifiesto respecto a la producción de *software* libre quizás sea la aportación principal de estas teorías.

Una analogía apresurada supone dos peligros interpretativos opuestos: por un lado, insistir en la reciprocidad y la generosidad como principios generales de la acción, opuestos a la producción capitalista de mercancías; por el otro, buscar bajo la apariencia del don la motivación no altruista. En el fondo, ambas perspectivas son complementarias, puesto que parten de ese dualismo entre don generoso y cálculo egoísta. Además de ser improductivo, asumir esta oposición nos haría caer en las paradojas de la motivación y de la imposibilidad de un don puro, imposibilidad analizada por Derrida (1995), puesto que siempre es posible adjudicar, o al menos sospechar, una motivación oculta ante un acto aparentemente desinteresado.

No hay que partir pues para explicar el *software* libre de una norma de reciprocidad abstracta y universal, que puede ayudar en el mejor de los casos a describir pero no explicar. Además, en realidad la cuestión de la reciprocidad no es cuantitativamente tan importante en este campo, en tanto son relativamente pocos los que en el ámbito del *software* libre aportan algo, en relación a los que lo utilizan.

Teniendo en cuenta todas estas precauciones, sí que puede resultar interesante y heurísticamente sugerente comparar el fenómeno del *software* libre con formas de don que se dan en economías precapitalistas. Al hacer esto, se insiste en el carácter de don del *software* en contraste con su carácter de mercancía en otros contextos.[2] Pero no hay que suponer que todo lo que ocurre en torno al *software*

[2] Es muy difícil estimar con realismo cuál sería el coste monetario de producir Debian como una mercancía. En 2012 el Desarrollador James E. Bromberger estimaba el costo del desarrollo de los

libre o la cultura libre es reciprocidad, cooperación, libre circulación de los bienes, y colaboración desinteresada, como por otra parte tampoco ocurre, por supuesto, en el *kula* o el *potlatch*. No nos podemos contentar con repetir ideas abstractas y generales sobre la colaboración, el compartir o la interdependencia. A fin de cuentas, todo esto no es lo específico de las «economías del don». Tampoco es lo más relevante para esta comparación el hecho de que nos situemos en un ámbito de «lo digital» que permitiría la copia y la distribución a coste cercano a cero, y cuyos bienes se caracterizarían por la no-rivalidad y la no-exclusión.

Lo que sí se tratará de hacer en este capítulo es utilizar algunos elementos de las teorías clásicas del don y sus desarrollos posteriores para poner de manifiesto algunas características de la producción y circulación del *software* libre. Una exploración de todas las posibles analogías y aplicaciones de las teorías antropológicas clásicas sobre el don nos llevaría demasiado lejos y nos desviaría del objetivo que se persigue en este trabajo. Lo específico de las economías del don que más nos permite iluminar los procesos que estamos considerando se situaría sobre todo en la imposibilidad de interrumpir la circulación de los bienes pertinentes, ligada al uso de las licencias, y en la existencia de una «obligación de recibir». Unido a todo ello, el papel concedido a los objetos y su circulación en la formación de vínculos sociales. Si vamos a considerar las teorías antropológicas sobre el don, lo haremos sin olvidar la perspectiva de la Teoría del Actor-Red. A fin de cuentas, ambas enfatizan la capacidad concedida a los objetos materiales para construir y consolidar relaciones y vínculos, enlaces sociales.

Partiremos entonces no de localizar las cuestiones que nos interesan en una «economía del don», como si el ámbito de producción y distribución del *software* libre fuese una instancia general de un tipo tal de economía, sino del don del *software* como hecho social total, en el sentido de Mauss de un fenómeno en el que se relacionan diferentes dimensiones, sin privilegiar ninguna de ellas. No se plantea en este capítulo que el *software* libre sea una «economía del don» (si es que tal cosa existe en absoluto), sino que se observa ese ámbito a través de esta metáfora. El código fuente, en tanto objeto que circula como «don» por las redes de las «co-

proyectos *upstream* en aproximadamente 19,070,177,727 dólares, probablemente estimando por lo bajo (Bromberger, 2012). Independientemente de que el cálculo sea más o menos exacto, da una idea de la dimensión de lo que supondría producir Debian como una mercancía.

munidades» del *software* libre, es un objeto peculiar cuya comprensión requiere ir más allá de, y poner en cuestión, los dualismos que venimos considerando. El código fuente, al menos el de los programas *libres*, es un objeto complejo que hace circular muchas cosas con él. Sus efectos se hacen sentir en las máquinas en las que funciona, pero también en las relaciones que permite establecer entre los individuos. Como hemos visto, permite conseguir determinados fines mediante su puesta en marcha, pero al mismo tiempo es una forma de expresión.

Más que entender el don como el hecho de dar sin expectativa de retorno, al nivel de las motivaciones, entonces, lo interesante es estudiar el mecanismo que mantiene los bienes en circulación. Estos mecanismos no sólo implican aspectos económicos, sino también jurídicos, tecnológicos o culturales. Lo fundamental son las relaciones que se establecen en torno al don y su objeto, en este caso al *software*. Como iremos viendo, determinados conceptos que en la explicación *emic* de las sociedades tradicionalmente concebidas como centradas en el don (y en la explicación de Mauss) dependen de elementos míticos o mágicos, tienen en el campo que nos ocupa correlatos empíricamente delimitables: del *hau* como espíritu del objeto a las licencias libres como fuerza que los mantiene en circulación, o de las genealogías míticas a la precisa atribución de *copyright* en los archivos del código fuente.

Aunque no es directamente aplicable en todos sus aspectos, un concepto que se ha mostrado particularmente revelador e inspirador es el de «posesiones inalienables», utilizado por Annette Weiner (1992) en su libro *Inalienable Possessions. The Paradox of Keeping-While-Giving*, y desarrollado posteriormente por Maurice Godelier (1998) en *El enigma del don*. Aunque no en el mismo sentido, la consideración del *software* libre simultáneamente como *don* y como *posesión inalienable* puede ayudar a comprender algunas de sus características. Del mismo modo, también se puede arrojar luz sobre algunas de las acciones sociales típicas del campo del *software* libre, especialmente Debian, considerándolas como un intento de resolver la paradoja expuesta por Weiner en su subtítulo: la paradoja de «guardar para (poder) donar», a la que habría que añadir con Godelier la de «donar para (poder) guardar».

El hilo conductor del presente capítulo será la consideración de las tres obli-

gaciones que constituyen el don según Mauss en este ámbito, especialmente las de *devolver* y *recibir*. Así resume Godelier esta cuestión:

> ¿Qué es lo que hace que en sociedades, épocas y contextos tan diferentes, los individuos y/o los grupos se sientan obligados, no solamente a donar o, cuando se les dona, a recibir, sino también se sientan obligados, cuando han recibido, a devolver lo que se les ha donado, y a devolver, ya sea la misma cosa (o su equivalente), ya sea alguna cosa mayor o mejor? (Godelier, 1998, pág. 23).

Por último se establecerá una relación entre esta última obligación y la prevalencia de la doocracia como forma de organización y gobierno. Para todo ello, lo primero es retomar la reflexión sobre lo que en este campo de acción funcionaría como objeto del don: el *software* libre.

5.1 El código fuente como objeto del don. ¿Por qué se dona?

5.1.1 Código fuente y don.

Para profundizar en la construcción de las relaciones que estructuran el campo de investigación en torno al fenómeno del «don», el primer paso es determinar cuáles son los objetos que funcionan como «dones». Como ya se ha dicho, más que partir en general de una norma de reciprocidad o una motivación generosa, es más explicativo partir del tipo de objeto que se dona, ver cuál es su especificidad y qué hacen los individuos con él.

En realidad, ésta es precisamente la cuestión que permite que las perspectivas de las teorías clásicas sobre el don y la Teoría del Actor-Red puedan complementarse en el estudio del *software* libre: la importancia concedida a los objetos y sus características concretas, no completamente disociadas de las personas. Igualmente, la importancia concedida al «espíritu» o *agencia* de las mismas y a su circulación para la constitución de relaciones sociales. Lo fundamental serían las

redes de las que estos objetos forman parte, las asociaciones que se establecen con ellos y a partir de ellos. No es que en los intercambios o circulaciones que pueden categorizarse como dones estas relaciones y asociaciones sean más importantes (también las mercancías juegan ese papel en un marco institucional determinado, el mercado), sino que son más visibles (por su menor separación de los sujetos) y nos permite tenerlas más en cuenta. Godelier señala, refiriéndose a la fuerza que se encarna en las tres obligaciones del don:

> Mauss describía esta fuerza como algo que dominaba a la vez a personas y a cosas, aunque por supuesto se producía en el seno de sociedades donde no parecía existir ninguna barrera absoluta entre unas y otras, donde no podían separarse radicalmente (Godelier, 1998, pág. 24).

Lo que podríamos matizar es que esas sociedades son exactamente todas las sociedades. No es algo propio de sociedades que todavía no hayan entrado en la modernidad. Es en ese sentido que Latour afirma que *Nunca fuimos modernos* (Latour, 2007).

Fundamentalmente, aunque no sólo, lo que funciona como don básico es el código fuente (véase la página 218) de los programas. Para comprender adecuadamente la naturaleza del código fuente como *don* hay que tener en cuenta que, históricamente, no es algo más «básico» o «fundamental» a partir del cual el programa binario sería algo derivado, sino que es el fruto de un proceso histórico de abstracción, algo que ha tenido que ser construido históricamente, y en una parte muy importante precisamente para que el *software pueda* ser compartido. Porque compartir *software* no es sólo compartir herramientas binarias listas para usar, sino el conocimiento que les subyace. Sólo compartiendo código fuente y no solamente binario se podrá éste acrecentar y devolver mejorado. En este proceso histórico, además del desarrollo de lenguajes de programación de alto nivel como C, COBOL, FORTRAN o LISP, es fundamental el papel del sistema operativo UNIX, sobre el que se basa GNU/Linux. Sobre esta cuestión véase la sección 4.1, *La Red como metáfora. Unix*.

Allí se puso de manifiesto la inseparabilidad de las diferentes dimensiones

que hemos identificado como constituyentes del *software* como objeto híbrido, y la cultura del compartir que UNIX ayudó a crear. Pero no sólo ha sido importante la separación entre código binario y código fuente, también la propia separación entre *software* y *hardware*. En realidad, ambos procesos están muy relacionados, puesto que el código binario depende en su forma concreta del tipo de máquina en la que funciona, mientras que el mismo código fuente puede ser compilado produciendo binarios diferentes capaces de funcionar en máquinas distintas. Fuller (2008a, págs. 2-3) señala el momento clave:

> Another crucial moment was the decision in 1968, prompted in no small part by antitrust court actions, to split its software section off from its hardware section. Software was no longer to be bundled as a service or gratuity. As a result, according to Martin Campbell-Kelly, "IBM liberated the industry by unbundling". At the point of software's legal reordering as a separate kind of entity, it became a commodity, an entity the prime or sole motive for the production of which is to generate a monetary profit [...]

Lo cual no deja de ser paradójico, puesto que para que el *software* pueda llegar a circular independientemente del *hardware* (y en el caso del *software* libre como don) primero ha de separarse del mismo, respecto al cual era una *gratuity*, es decir una propina o gratificación, pero también un *gift*, un don, un regalo. Las empresas no vendían *software*, vendían máquinas y ofrecían el *software* como algo de más, añadido. La obligación de separar ambas cosas abre la posibilidad de comercializar el *software* por separado. Y es la comercialización del *software*, su producción para ser vendido, lo que a su vez permite que se produzca esa distinción entre *software* como mercancía y como don, que con anterioridad no tenía sentido. Pero esta separación es inestable, como se ve en las posibilidades de obtener ganancias produciendo, distribuyendo o dando soporte al *software* libre. En cualquier caso, un elemento fundamental en el proceso de aparición de lo que hoy conocemos por código fuente es el objetivo de hacer el *software* portable, capaz de circular y funcionar en diferentes máquinas y situaciones, desligándolo lo más posible de una asociación demasiado estrecha con determinado *hardware*, lo que limitaría radicalmente su capacidad de traslación.

En el caso de Debian, la cuestión del «don» ha de abordarse además teniendo en cuenta su papel como *proxy* (véase la sección 4.6.1, *Debian como* Proxy. Upstream *y* downstream) en la circulación de *software*, al ser una *distribución* que pone a disposición de los usuarios una colección de *software* en su mayor parte creada por otros individuos o grupos. Así, en Debian se produce una circulación y una transformación de los objetos que se donan. Lo que se hace con ellos en Debian es empaquetarlos, integrarlos y distribuirlos, aportando un valor añadido. Si el fenómeno del don cuando hablamos de *software* libre se refiere al código fuente, en el caso de Debian lo que se dona más específicamente es su empaquetado e integración con un sistema completo.

5.1.2 ¿Por qué se dona el *software* libre?

¿Por qué los desarrolladores de *software* libre ofrecen su creación gratuitamente, como un regalo? ¿Por qué los miembros de Debian dedican tantas horas y energías a una actividad por la que no son recompensados económicamente? ¿Por qué no lo producen como mercancía? Si partimos de la oposición entre don y mercancía, y usamos solo categorías económicas, efectivamente la cuestión aparece como un enigma. Y el tipo de explicación más extendida consiste entonces en buscar otros bienes con un valor económico indirecto, como el prestigio o la reputación. Contribuir a proyectos de *software* libre es una forma de aumentar el propio capital social y cultural (Bourdieu, 1977), que luego podrá ser convertido en capital económico.[3] También es común asimilar esta labor al voluntariado o el activismo, donde lo más importante sería un sentimiento de altruismo, generosidad, o incluso reciprocidad.

Estas explicaciones pueden tener mucho de pertinente y acertado, pero no tienen en cuenta uno de los puntos centrales de la cuestión, el tipo de objeto específico que funciona como don. Un objeto inmaterial[4] sujeto al régimen de propiedad intelectual, donde lo que está en juego es la relación entre lo que se da y lo que no se da, entre lo alienable y lo inalienable. Hay aquí dos cuestiones diferentes: por qué el *software* libre se constituye y mantiene como don, o quizás mejor

[3] Un claro ejemplo de este tipo de argumentación puede verse en Matzan (2004).

[4] Como señala Godelier, las cosas que se donan no son solo objetos materiales, también se incluyen danzas, actos de magia, apoyos en conflictos, etc: «podríamos decir que ese dominio está constituido por *todo* aquello cuya *distribución* es posible, *tiene sentido* y puede crear obligaciones, o una deuda, en el otro» (Godelier, 1998, pág. 150).

expresado, qué ocurre cuando se dona código; y por qué se contribuye con tiempo y esfuerzo al desarrollo de la distribución, lo que en muchos casos no implica donar código. Vamos a verlas por separado.

Respecto a la primera cuestión, el *software* libre se puede entender como un caso paradigmático de la expresión usada por Weiner (1992), «posesión inalienable». Y su desarrollo como una elaboración de la paradoja que plantea en el título de su obra: la paradoja de *Keeping-while-Giving*, «Guardar para (poder) donar». Se trata de explicar las reglas que rigen el don a partir del intento de solucionar esta paradoja. Para ello hay que determinar qué tipo de posesiones se tratan de mantener fuera de la circulación. Estas serían las *posesiones inalienables*. La tesis central de Weiner es que hay cosas que no se deben donar, que no deben circular, entre otras cosas para que sea posible el intercambio de las demás.

Y esto es precisamente lo que caracteriza a los objetos que son licenciados con una licencia libre: que para que puedan circular de la manera en que lo hacen, ser compartidos, hay algo que de alguna manera no se cede, precisamente la propiedad intelectual, el *copyright* de la obra. Se cede, se hace circular, el derecho a usar la obra, a modificarla, a transmitirla; lo que nunca se cede es precisamente el derecho a retirar esa obra y sus modificaciones y mejoras de la circulación. Licenciar una obra con la GPL u otra licencia libre (al menos con las licencias *copyleft*; véase la distinción más adelante) supone convertirla precisamente en bien inalienable, dificultando su privatización. Supone en definitiva resolver la paradoja planteada por Weiner.

En realidad, por supuesto, el *copyright*, los derechos comerciales de copia y explotación, sí son alienables y se pueden ceder o vender a una empresa. Lo que se hace con las licencias libres es precisamente evitar esa posibilidad de apropiación privada: una vez que la obra ha sido licenciada con una licencia libre, no puede ser retirada de la circulación (sí una versión posterior suya, pero no la publicada en ese momento). Si tiene sentido decir que son inalienables, es porque una vez publicadas no pueden ser alienadas de quien posee ahora los derechos a usarlas, no pueden ser sacadas del ámbito del procomún, como enseguida veremos. Al donar una obra al procomún, al licenciarla con una licencia libre, se asegura que se mantendrá en el procomún. Si se usa una licencia propietaria y el autor

aliena sus derechos comerciales, nada le asegura que esa obra siga de alguna manera circulando. Al poner su obra en el procomún,[5] se asegura que se guarda en éste: *Keeping-while-Giving*. Lo que cambia es de alguna manera el sujeto de esa posesión, al menos de los derechos de uso, transformación o distribución: del autor del código pasan a un grupo mayor, que puede incluir a cualquiera. Aquí se encuentra posiblemente la mayor diferencia entre los fenómenos estudiados por Mauss, Weiner o Godelier con lo que ocurre en el *software* libre. Pero las posesiones inalienables analizadas por Weiner lo son también de un grupo o colectividad, aunque en esos casos sea más restringido. Señalaremos dos cosas: el hecho de que el *software* sea un bien digital, reproducible, lo distingue esencialmente de otros en que no es necesario excluir de su uso o propiedad a otros para conservarlos para uno mismo. En una sección posterior consideraremos estas diferencias entre los bienes comunes tradicionales y los digitales. La segunda es que aunque de hecho nadie puede ser excluido del uso y disfrute de los bienes que se encuentran en el procomún, sí que las corporaciones que desarrollan, integran o usan *software* privativo pueden verse excluidas de integrar los bienes comunes digitales en sus productos (al menos si los quieren mantener como privativos).[6]

Estos derechos comerciales y de distribución se basan además en otros derechos que son más propiamente inalienables: los derechos morales del autor. Básicamente, el derecho de atribución de la obra y el derecho a que se respete la integridad de la misma. Con mayor o menor grado según las diferentes legislaciones nacionales, quedan recogidos en la Convención de Berna, y protegen a los autores de obras creativas (de nuevo, esto depende de las legislaciones nacionales; en algunas (Europa) se reconocen para las creaciones de *software* y en otras no (EE.UU.)).[7] Incluso cuando los derechos comerciales se ceden (bien a otro particular, bien al procomún), siguen existiendo una serie de derechos del autor a los que no se puede renunciar. La creación (artística o de *software*) establece una

[5] No en el dominio público, véase la sección 5.2.1, *Las licencias libres*.

[6] A no ser que se use un esquema de licenciamiento dual, en el que se ofrece un *software* con una licencia *copyleft*, pero se ofrece también paralelamente la posibilidad de adquirir ese *software* con una licencia propietaria, pagando por ella. En ese caso, sí podrían quedar excluidas de la circulación las modificaciones y obras derivadas que realizase quien adquiera esa licencia. Se puede producir así una apropiación de los cambios realizados a ese *software* libre, como puede ocurrir también al usar una licencia libre no *copyleft*, como veremos más adelante. Aquí analizamos el caso general.

[7] Véanse Van den Brande *et al.* (2014) y Sundara Rajan (2011, pág. 508).

relación entre obra y autor que no se puede romper. Más allá de la legislación concreta vigente en cada país, lo fundamental aquí es el hecho de que el acto creativo difumina la distinción entre el objeto donado y el donante.[8] Así lo explica Godelier respecto al intercambio de dones entre los baruya:

> Si el contradón no elimina la deuda, es porque la «cosa» donada no se ha separado, no se ha escindido completamente del que la dona. La cosa ha sido *donada, sin ser verdaderamente "alienada"* de quien la dona.
>
> Así, la cosa donada arrastra consigo algo que forma parte del ser, de la identidad de quien la dona. Pero hay más, en la medida en que el donante no deja de tener derechos sobre la cosa tras haberla donado. [...] donar significa transferir sin alienar o, por emplear el lenguaje jurídico propio de Occidente, donar supone ceder los derechos de uso sin ceder por ello el derecho de propiedad (Godelier, 1998, pág. 67).

En cualquier caso, ésta es una relación que pretende ser protegida por las licencias libres incluso cuando no está bien establecida por las legislaciones nacionales, como las de EE.UU. Volveremos sobre esto al hablar sobre el procomún y la obligación de devolver.

Pero en Debian no es la autoría del código o el trabajo de empaquetar *software* lo único que se mantiene como posesión inalienable. También están, por ejemplo, la marca «Debian» o su logo oficial (Debian.org, s.f.[r]). Al ser *software* libre, es perfectamente posible volver a distribuir sus paquetes o incluso crear una nueva distribución basada en ella (como vimos en la sección 4.6.1, *Debian como Proxy. Upstream y downstream*) sin que sea necesario pedir permiso. Lo que no es posible es decir en este caso que lo que estás distribuyendo es Debian sin más. Sería necesario cambiarle el nombre, que permanece inalienable, y en este caso en relación a un grupo concreto. Inalienable en el sentido en que el grupo se esfuerza para que permanezca como su propiedad.

[8] Idea desarrollada por Strathern (1990, 2004).

Ya hemos hablado también de la importancia que tiene en Debian la *ownership* o *maintainership* de los paquetes, en el sentido de quién tiene el derecho a mantenerlos para la distribución. También del acceso a los recursos, como los servidores del proyecto. Estos derechos y posibilidad de acceso pueden pasar de un desarrollador a otro, pero como hemos estado viendo, muchas veces con dificultad y sujeto a una negociación. El carácter inalienable de estos bienes, como afirma Weiner, está siempre en disputa y expuesto al riesgo.

¿Quién es entonces el propietario de esos bienes inalienables? Si hablamos del código, ciertamente hay un propietario, el del *copyright*. Si hablamos de otros elementos como la marca, el Proyecto en su conjunto y su representación legal. No es sólo un individuo. Pero hay también un sentido en el que son posesiones inalienables en relación a una «comunidad», en el caso del código definida amplia y difusamente. No es ni tan siquiera un grupo específico quien tiene esa relación. Para determinar en qué sentido y extensión ocurre hablaremos en una sección posterior del *procomún*. Sólo a partir de ahí se puede entender en qué sentido el *software* libre es propiamente inalienable. Y es que el procomún, el *software* libre, está a disposición de todos, aunque sean grupos específicos los que lo distribuyen, con obligaciones y derechos específicos. Es inalienable porque nadie puede perder su acceso al mismo, no porque se pueda excluir a otros.

En cuanto a la segunda cuestión, por qué se contribuye al proyecto con tiempo y esfuerzo, lo primero que hay que señalar es que generalmente esto está muy relacionado con la obligación de devolver. Lo más usual es que quien se plantea contribuir de alguna manera siente que ya ha recibido mucho, y se trata más bien de devolver. No puede ser de otra manera puesto que a quien se da no es un individuo, sino el procomún o la comunidad del *software* libre. Al estar ésta ya preconstituida, una donación particular es siempre de alguna manera una devolución.

Un ejemplo nos lo proporciona el mismo creador de Debian, Ian Murdock, en una intervención en una mesa redonda sobre la historia de Debian en Debconf 4, grabada y publicada por Coleman (s.f.):

Ian – What really grabbed me off the bat was the community. That was

> what really grabbed me and you have to understand at the time, it was a completely foreign notion that I somehow I had stumbled upon this group of people that were interested in the same things that I was interested in who had basically, for no particular reason built this thing, this operating system and it had actually worked and I could do my work on it and I had not paid a dime for it, they did not ask anything of me when I download it or used it.
>
> And whenever you are in a situation like that when people have given so much to you, one of the first instincts is like: "what can I do for you, what can I give back?"

Además, en el caso del código encontramos una ilustración literal de la idea desarrollada por Mauss y la tradición antropológica que le siguió: la inseparabilidad de la persona y el objeto que dona. En efecto, y como hemos visto, el nombre del autor (también del empaquetador, el traductor, o el de cualquiera que haya colaborado en él por ejemplo enviando un parche) está incluido en el código fuente del programa, o al menos queda constancia en los repositorios de *software* que alojan esos proyectos. Si la única posibilidad de esa entelequia que es el don puro basado en el altruismo reside en su anonimato, el *software* libre no lo es en la inmensa mayoría de los casos, puesto que no es anónimo, aparece siempre ligado a su autor. Eso es lo que hace que pueda ser usado para conseguir otros bienes como el prestigio o un empleo. El autor dona el código, pero mantiene para sí muchos de los beneficios de su creación. También veremos que es una condición que permite superar el proceso para ser admitido en el Proyecto como Desarrollador de Debian.

En un *post* celebrando sus diez años como Desarrollador de Debian, un Desarrollador muestra la estrecha relación entre los motivos del deseo de devolver y contribuir, el interés profesional y la pertenencia a la comunidad:

> I've believed in Debian's values since my classmate Ulisses Alonso prodded me to install Debian on my desktop back in 1997. Even if getting X up and running on bo was a real pain in the ass, knowing that the

> system I was running had all been written by people driven by altruism was enlightening; months later it was time to give back. [...]
>
> I am very proud of having been a part of an incredible project like Debian, and hope to be around for at least ten more years. Not only because I love and believe in Free Software; thanks to my involvement, I've been able to work on Debian-related jobs for all of my professional career, but above all I've been very lucky to make lots of real friends (Mallach, 2009).

Y vemos motivaciones parecidas en varias de las entrevistas realizadas:

> **gwolf:** Desde que empecé a utilizar *software* libre sentía la necesidad de contribuir de vuelta. Todo esto lo estoy recibiendo, ¿cómo voy a contribuir con ello?

> **rmayorga:** Es lógico creo yo si eres usuario. Por años, uno siente que tiene que regresar todo lo que ha venido captando. Pura lógica, porque estoy tomando todo de la comunidad y no estoy dando nada. Es el simple hecho de sentir que ya suficiente tiempo viví de los que otros hacen. Hoy quiero hacer yo, para que otros tengan.

También en el siguiente fragmento de entrevista a una Desarrolladora se observa esa reacción casi automática y vivida como evidente de querer contribuir de vuelta:

> **amaya:** Una de las cosas más bonitas del *software* libre es cuando empiezas a obtener, sin haber puesto tú nada inicialmente, un sistema operativo que funciona, y una comunidad de usuarios que están encantados de ayudarte, no sólo a que funcione mejor sino a que entiendas lo que está pasando y por qué no te funciona bien. Te dan pescado y te enseñan a pescar. Enseguida quieres tú ponerte a ayudar también, a

compartir lo que has aprendido. Ponerte a ayudar es automático, empiezas a querer ayudar enseguida.

O de manera más extensa:

agi: Te metes en Linux porque es el que te puedes bajar de Internet, con el que puedes cacharrear, y luego ya empiezas además a ver la filosofía, que aquí la gente aporta, y te lo da para que hagas con él lo que quieras. Y es más, si quieres aportar tú se lo puedes mandar de vuelta y hay una comunidad que se ayudan entre ellos. [...] pues empezó cuando salí de la gran consultora y conseguí entrar en una empresa de consultoría de Linux [...] que ahí es realmente donde empecé a aprender a un ritmo grande, pues vi que yo me estaba beneficiando enormemente de un montón de trabajo que habían hecho otros, y que si todo el mundo hiciera lo mismo sin aportar un poco de vuelta, los otros se cansarían y lo dejarían. Entonces para mí, y eso me lo dijo una persona muy claro, si todos sacamos peces y ninguno ayudamos a meter algo, pues aquello se seca. Para mí siempre ha sido muy claro, yo me estoy beneficiando de un montón de trabajo y conocimiento de otras personas. O yo aporto mi modesta aportación por mínima que sea, o aquello se secará.

Aquí vemos además insinuada la idea del procomún tal como la trataremos más adelante. En cualquier caso, dar o devolver supone básicamente entrar en una red de relaciones. Es más importante esto para entender la acción social y las redes sociotécnicas que se constituyen, que la motivación individual. De hecho, puede considerarse como la motivación más importante, se entienda como una manera de posicionarse en el mercado de trabajo, contribuir al procomún o entrar en la comunidad.

Así, cuando en las entrevistas se habla de la motivación (para donar código, tiempo, o cualquier otra cosa) que supone el sentimiento de «devolver algo a la comunidad», el deseo que se manifiesta es más el de formar parte de la comunidad que el de devolver una deuda. Se entiende la propia actividad, muchas

veces incluida la laboral, como una contribución a un repositorio común (del que enseguida hablaremos) del que extraen lo que necesitan para sus prácticas. No se trata aquí de una lógica de la mercancía y de la deuda, sino de la constitución de una «economía humana» en el sentido de Graeber (2011): lo importante es la producción de relaciones sociales y seres humanos. La producción de una comunidad. Seguiremos tratando esta cuestión de la relación entre comunidad y don en la sección 6.1, *De la comunidad al colectivo sociotécnico*.

Citaremos dos motivaciones más para acabar. En primer lugar, el papel que juegan el placer y la diversión. Como dice Graeber (2011, pág. 99), «sharing is not simply about morality, but also about pleasure». O como lo expresan unos Desarrolladores de Debian:

> **sez:** If you volunteer for something, you do it because it is fun. And if a team atmosphere is not fun, then you don't want to do it in your free time. That's what you do when you are paid, because you have to. If you don't have to, and it's not fun, why do it?

Y, añadiendo además toda una lista de motivaciones para colaborar en la comunidad del *software* libre:

> **dato:** Mucha gente lo hace por *hobby*, porque le gusta y porque lo disfruta. Yo odio cuando veo acciones de otros desarrolladores o mías en las que noto que a otra persona se le está robando la diversión, eso me parece algo horrible y que hay que evitar a toda costa. Cuando a alguien se le roba la diversión es un contribuidor que estás perdiendo. [...]
>
> La palabra diversión viene de una frase de Torvalds «it's all about the fun» [...]. Yo más que diversión creo que es el disfrute, el placer de hacer las tareas que tú has elegido voluntariamente, que tú has decidido emplear tu tiempo en eso y a cambio te proporcionan un disfrute. Si estás *debuggeando*, no es la tarea más divertida, pero se puede disfrutar, yo creo que al final se hace porque al final del día te quedas con un

> buen sabor de boca. Para mí, al empezar a usar *software* libre, las motivaciones para colaborar, que no las he tenido siempre claras desde el principio, sino que las he ido descubriendo, lo primero es por gratitud, este sistema es una maravilla y puedo colaborar, intentar que sea aún mejor; la siguiente que yo encontré fue recibir *feedback* de que mis colaboraciones eran útiles; una tercera es poder trabajar con personas muy competentes de las que he podido aprender técnicamente mucho [...]; y una cuarta es el aspecto social, te empiezan a unir lazos más que meramente técnicos con otras personas. Algunas personas se han convertido en amigos.

Y muy ligado con esto, la posibilidad de adquirir conocimiento. Mencionaremos sólo un testimonio en este sentido:

> **guillem:** Una de las cosas que me ha atraído del *software* libre ha sido [...] el tema de poder encontrar información muy detallada de cómo funcionaba el sistema libremente. Y el hecho de aplicar eso también al código fuente, y al proyecto en sí, era algo que me interesaba. Por el tema del conocimiento. Y aparte también el poder devolver a la comunidad. [...]
>
> Yo creo que en ese momento era más importante el hecho de poder aprender, y era como que tenías a tu disponibilidad un grupo de gente que tenía muchos conocimientos. Era como un núcleo de gente con conocimientos muy altos que podías entrar y absorber eso. [...] Yo supongo que en ese momento fue más por el tema del conocimiento y por el hecho de tener una oportunidad de aprender mucho.

5.1.3 Trabajo voluntario y trabajo remunerado: tensiones y controversias.

En la producción de *software* libre, las lógicas de la producción de mercancías y la del don no son excluyentes sino complementarias, como muestra por ejemplo Berdou (2011), y son frecuentes las situaciones en las que ambas se combinan, así como aquéllas en las que se puede transformar un tipo de capital en otro.

Así, algunos desarrolladores empaquetan porque es útil para su negocio o porque alguna empresa les paga para ello. El trabajo voluntario en un proyecto como Debian también se convierte muchas veces en la mejor manera de demostrar las propias habilidades como informático, y atraer así el interés de los reclutadores de grandes empresas del sector. Es relativamente fácil pasar de una lógica a otra, como muestra por ejemplo Gregory (1997), pero ambas mantienen su especificidad, sin que sea posible reducir sin más el don a una forma de capital o de mercancía.

Pero el intento de establecer, o simplemente considerar, alguna forma de remuneración más o menos directa por el trabajo voluntario en el seno del proyecto Debian, es una fuente potencial de tensiones y conflictos, fundamentalmente por el daño que la lógica del trabajo remunerado puede hacer a la lógica del trabajo voluntario. La transformación explícita de un tipo de valor en otro, la comunicación de ambas esferas de valor, se ha mostrado entonces como problemática. En esta sección no se discuten todos los casos en que se producen intercambios entre empresas y Debian (como cuando diferentes empresas donan *hardware* o contribuyen a financiar conferencias y reuniones presenciales, o desarrollan una distribución basada en Debian como Ubuntu),[9] sino que nos limitaremos a considerar algunos casos que implican la aparición de trabajo remunerado. Tampoco entraremos en detalles sobre la forma en que Debian gestiona sus recursos monetarios a través de sus *Trusted Organizations* (Wiki.debian.org, s.f.[af],[h]).

Es posible aparecer como consultor (alguien que proporciona soporte o servicios relacionados con sistemas Debian a cambio de dinero) en la página web del Proyecto Debian (Debian.org, s.f.[s],[t]), aunque esto no indica una garantía ni recomendación oficial por parte del Proyecto. Algunos Desarrolladores aparecen aquí, si bien las normas impiden su identificación inmediata como miembros del Proyecto frente a otros consultores, al no ser posible usar como dirección de correo de contacto la propia del proyecto (acabada en `@debian.org`) o una dirección bajo el dominio oficial `*.debian.org`.

[9] Como señala Hill (2012), estos procesos pueden tener sus propios problemas, pero limitados si se mantiene la transparencia y la independencia del Proyecto, como ocurre en Debian. Una visión general sobre el uso que Debian hace de sus fondos y sobre las cuestiones financieras puede verse en Wiki.debian.org (s.f.[ae],[ad],[ac]).

Por otra parte, el proyecto Debian LTS (*Long Term Support*),[10] iniciado en 2014, pretende prolongar el periodo durante el que las diferentes versiones de Debian pueden ser usadas de forma segura, básicamente recibiendo actualizaciones de seguridad, más allá de periodo cubierto por el equipo de seguridad, hasta un mínimo de 5 años. Pero este trabajo es impulsado no sólo por voluntarios, sino también por diferentes compañías a través del pago a una serie de Desarrolladores y Mantenedores de Debian (todos pueden participar si están interesados, independientemente de que reciban o no algún pago). Estos pagos se realizan sobre todo a través de la compañía Freexian (Freexian.com, s.f.[a]), que coordina los pagos realizados por diferentes compañías interesadas y el reparto de las diferentes tareas entre los contribuyentes pagados. Los informes mensuales sobre los trabajos realizados y las compañías que contribuyen se pueden consultar públicamente (Hertzog, s.f.). Los beneficios para las compañías que contribuyen con financiación son claros: pueden influir en que se dedique más esfuerzo en los paquetes necesarios para ellas, o tener comunicación directa con el proyecto LTS (Freexian.com, s.f.[b]).

Como en los casos en que una compañía dedica el tiempo de alguno de sus trabajadores a trabajar en paquetes de Debian, esta iniciativa no ha suscitado prácticamente ninguna controversia, puesto que el dinero no proviene de los beneficios o ganancias que pudiera obtener el propio Proyecto Debian. Pero ésta si ha aparecido cuando se ha propuesto que sea el mismo Proyecto Debian quien pague a voluntarios con dinero recaudado por Debian, o lo hagan empresas directamente, por realizar tareas esenciales para el desarrollo de la distribución.

Hill (2012) analiza algunos de los problemas que se producen cuando una compañía pretende invertir dinero en algún proyecto voluntario relacionado con el *software* libre, usualmente pagando a desarrolladores, señalando incluso el resultado paradójico de que «a menudo es más fácil conseguir dinero para un proyecto voluntario de *software* libre que gastarlo». Entre los casos y las estrategias posibles analizados hay algunos relativos a Debian.

Estos problemas incluyen la disminución del número de voluntarios y de

[10] Se puede acceder en Wiki.debian.org (s.f.[ag]). Una discusión del proyecto y balance de sus dos primeros años puede verse en una charla de su principal impulsor, en Debconf16.debconf.org (s.f.[a]).

su grado de implicación, una disminución de la transparencia (debido quizás simplemente a la posible mayor cercanía y comunicación de los trabajadores pagados, que pueden incluso compartir oficina), o la excesiva influencia de los intereses de los empleadores en la dirección que toma el proyecto por encima de las del resto de usuarios (Hill, 2012). Sobre esta cuestión se celebró también en DebConf 16 el BoF[11] «Using Debian Money to Fund Debian Projects» (Debconf16.debconf.org, s.f.[b]), inspirado en buena parte en el artículo de Hill y conducido por el Desarrollador que está detrás de Freexian y DebianLTS.

Una visión muy crítica al respecto de todas estas cuestiones la podemos encontrar en la siguiente entrevista con un Desarrollador de Debian:

> **clint:** The way I believe that free software worked was that people would just give. They would make changes for themselves, but they would give them to everyone, and this would make life better for everyone. And then, because they would be doing that, and their neighbors would be doing that, they would also get the benefits of ..., its gifts. So it's not selfless but it is ..., everyone giving and then receiving all the benefits together. There lies the beauty of Debian because it's not about greed, and I think that when greed touches free software is a very ugly thing. And I think we are in danger because it is becoming a popular idea to have micro payments, or Paypal, or people soliciting donations. Before, someone might say, "I would really like this feature, can you help me?", and someone might just write that feature for them, because they wanted their software to be better, they are proud of their software, and want more people to be able to use it. Now you hear people say, "well, if you pay me, I'll do that" [...] and I think that could destroy the beauty I see in free software. [...]
>
> When people start getting paid their priorities change. It is not just the Developers refusing to write new features that people want unless they are paid for them, it is also them not accepting patches that other people send. And I think that it breaks down the foundations of why free software works.

[11] Una reunión informal. Véase la sección 6.2.1, DebConfs.

El caso más conocido y que más controversia ha creado en el Proyecto Debian fue el proyecto Dunc-Tank.[12] En septiembre del año 2006 se constituyó un grupo de Desarrolladores (que incluía al Líder del Proyecto Anthony Towns y a algunos Desarrolladores muy activos y respetados en la comunidad) cuyo objetivo era financiar a dos *Release Managers* a tiempo completo durante un mes para conseguir que la siguiente versión del sistema operativo (*Etch*) se publicase a tiempo. Se pretendía que funcionase también como un experimento a pequeña escala sobre la posibilidad y deseabilidad de recompensar económicamente a Desarrolladores por su trabajo en determinadas áreas del Proyecto. La principal diferencia con los ejemplos vistos anteriormente consiste en que en este caso se trataba de pagar a Desarrolladores por actividades que entraban dentro de sus tareas y responsabilidades como miembros del Proyecto y del *Release Team*.

El dinero se consiguió y se financió a los dos *Release Managers*, que pudieron aumentar así la cantidad de tiempo y trabajo dedicado al Proyecto. Sin embargo, al final *Etch* se publicó en mayo de 2007 en lugar de la fecha prevista de diciembre de 2006.[13] Aunque es difícil evaluar su impacto real, en parte este nuevo retraso pudo deberse a las controversias producidas, que incluían el abandono del mantenimiento de paquetes por parte de algunos Desarrolladores, llegando algunos de ellos a abandonar el Proyecto. Además se produjeron discusiones muy acaloradas en las listas de correo y en *posts* públicos, e incluso Resoluciones Generales de apoyo y de rechazo al Líder del Proyecto. Los principales problemas mencionados eran la desigualdad que podía introducir en el Proyecto, así como la posible desmotivación de los voluntarios, disminuyendo así la cantidad global de trabajo dedicado al proyecto. En cualquier caso, fue un asunto que dividió mucho al Proyecto, y se siguió discutiendo durante mucho tiempo. No hubo acuerdo sobre hasta qué punto el experimento fue un éxito o un fracaso, pero no se volvió a repetir.

Sin entrar a valorar el impacto real posterior de todas estas cuestiones en

[12] Algunas referencias interesantes para una visión general de esta cuestión son Tay (2007) (entrevista con Anthony Towns, líder del Proyecto en aquel momento), ris (2006), Byfield (2006, 2007), o los enlaces que se encuentran en Wiki.debian.org (s.f.[ah]).

[13] Este retraso no es inhabitual en Debian, y en parte fue producido también por la necesidad de eliminar el *firmware* no libre. Véase la sección 2.2.4, *Un ejemplo: el* Firmware *y la inscripción de programas de acción*.

la marcha del Proyecto, lo cierto es que a lo largo del trabajo de campo realizado la mención a esta controversia pasada ha sido constante en diferentes entrevistas realizadas y en la documentación analizada, mostrando que la cuestión de fondo (la interacción de las lógicas del trabajo remunerado y del trabajo voluntario) sigue siendo una cuestión central para entender la dinámica del desarrollo de *software* libre.

5.2 La constitución del procomún a través de las licencias libres. La obligación de devolver.

Ya hemos visto algunas de las razones individuales que los sujetos alegan para devolver, es decir, para donar. Veamos ahora la obligación de devolver desde un punto de vista más estructural.

De las tres preguntas que se planteaba Mauss sobre el don, la más extensamente tratada y discutida ha sido la relativa a la obligación de devolver, entre otras cosas por la crítica de Lévi-Strauss al planteamiento de Mauss. Como es bien conocido, Lévi-Strauss reprochaba a Mauss haberse dejado mistificar por el pensamiento nativo y el concepto Maorí de *hau* como la fuerza que obligaba a devolver la cosa donada (sobre esta cuestión, véanse Godelier (1998), Lévi-Strauss (1971), Weiner (1992), Sahlins (1997)). Éste era básicamente para Mauss el enigma del don, la existencia de la obligación de devolver, que sería «la más importante en la práctica y en teoría como la más difícil de comprender» (Godelier, 1998, pág. 29). Si esta obligación existe es porque la cosa recibida no es algo inerte, sino que tiene espíritu (*hau*). Sobre la fuerza que obliga a devolver, Sahlins resume así la explicación de Mauss: «The *hau* is that force. Not only is it the spirit of the *foyer* but of the donor of the gift» (Sahlins, 1997, pág. 70). Es el espíritu de la cosa donada (*hau*), relacionado con el donante, el que obliga a devolver. La solución de Lévi-Strauss a este enigma tampoco resultaría útil, en tanto que se limita a postular un principio básico y universal de reciprocidad que da por supuesto aquello que hay que explicar. No es el momento de repetir una discusión tan extensa y con tantos desarrollos, pero sí conviene señalar algunos de sus aspectos.

Efectivamente, esta pregunta sobre la obligación de devolver es la que señala al mecanismo fundamental del don, también en el caso del *software* libre. Efectivamente, aquí es una misma fuerza o mecanismo lo que se encarna en tres obligaciones, distintas pero encadenadas. Y efectivamente, con Mauss y contra Lévi-Strauss, esto es así porque la cosa donada está animada, y no en virtud de que sea expresión de una reciprocidad universal. Por el contrario, en el caso del *software* libre aparece como la más fácil de comprender.

En este campo, esta fuerza que obliga a devolver es una fuerza legal, son las licencias que se aplican a los objetos donados, el código fuente de los programas. Esas licencias son puestas por el autor del código, quien tiene el *copyright*. Como el don descrito por Mauss lleva consigo el *hau*, el espíritu del donante, el código lleva el *copyright* y la licencia. Y los lleva consigo literalmente, en tanto que son parte del paquete fuente (véase la sección 4.5, *El empaquetado como práctica sociotécnica*). En Debian esto es un requisito indispensable para que un paquete pueda entrar en el archivo.

Éste es el mecanismo dominante que hace que tenga sentido preguntarse por el *software* libre como don. Si la obligación de dar o de recibir son siempre relativas, la de devolver en el sentido especificado por las licencias libres es absoluta, no se puede incumplir sin arriesgarse a sanciones legales. Además, las demás formas de don (tiempo, esfuerzo) que se han mencionado dependen de ésta, y de cómo contribuyen a la constitución de un procomún. Se contribuye a un procomún que se ha formado básicamente en torno a objetos licenciados, incluso aunque no se haga creando más objetos del mismo tipo.

Analizando este mecanismo podemos evitarnos la dificultad señalada por Graeber (2001) respecto a la vaguedad de muchos de los términos usados por Mauss, Lévi-Strauss y otros teóricos del don:

> The problem is always one of finding viable terms of comparison, and in this case I think the problem is particularly acute. Mauss' own terminology –the "potlatch," the "total prestation," "the gift," "reciprocity"– served well enough for making broad moral points about the logic of

> the market, but as terms of cross-cultural comparison, they are blunt instruments: extremely imprecise. In fact, I would argue that the muddiness of his terms made it impossible for him to frame his basic questions –particularly, "Why is that gifts have to be repaid?"– in a way that they could be meaningfully answered. Not that Levi-Strauss (1950) did much better: in fact, the term he fixed on to solve the problem, "reciprocity," is really the bluntest instrument of all. As currently used, "reciprocity" can mean almost anything. It is very close to meaningless (Graeber, 2001, pág. 217).

5.2.1 Las licencias libres.

Una licencia es un dispositivo legal que determina las condiciones de uso y distribución de un programa informático. La licencia libre más extendida y usada es la *General Public License* (Gnu.org, s.f.[b]), que establece la obligación de no redistribuir el programa, o una modificación del mismo, sin conceder al usuario las mismas libertades. Es decir, prohíbe su apropiación privada, originaria o derivada. Pero puede prohibir esto porque las leyes que regulan el *copyright*, pensadas para proteger la propiedad intelectual, le permiten hacerlo.

De hecho, lo único que define que un determinado *software* sea libre es la licencia que se le aplica. En Debian se recoge lo que una licencia debe permitir para que el *software* cubierto por ella sea reconocido como libre por el proyecto en un documento llamado *Debian Free Software Guidelines*.[14] Éstas son además la base de la definición de la *Open Source Initiative* (la veremos en el siguiente apartado, en el marco de la distinción entre *free software* y *open source software*), lo que hace que su relevancia práctica en el mundo del *software* libre sea muy elevada.

¿Qué significa que las licencias libres establecen una obligación de devolver? Ante todo, debemos distinguir entre licencias libres *copyleft*, como la GPL,[15] y licencias libres no *copyleft*, como la *BSD License* o la *MIT License*.[16] La diferencia

[14] Véase el apéndice B, *DFSG*. Puede encontrarse una amplia e interesante discusión sobre algunas implicaciones y como interpretarlas en «DFSG and Software License FAQ (Draft)» (Pearlmutter, s.f.).

[15] Una exposición detallada y práctica del funcionamiento de las licencias *copyleft* y sobre todo de la GPL puede encontrarse en Kuhn (2015).

[16] Son también muy utilizadas para otro tipo de obras, no de *software*, las licencias *Creative Com-*

fundamental entre ambas es que las *copyleft* exigen la publicación[17] de los cambios producidos al *software* con una licencia también libre (se habla así del carácter «vírico» de la GPL). Las licencias *copyleft* establecen así una obligación más fuerte de devolver. Hay que entender esta obligación como la imposibilidad de privatizar los objetos que han sido donados mediante su licenciamiento, de sacarlos de la circulación a través de diferentes espacios y convertirlos en mercancías de cuyo uso se puede privar a los demás. Ahí está la diferencia fundamental entre licenciar una obra sobre la que se tiene el *copyright* con una licencia *copyleft*, y hacerlo con una que no lo sea.[18] Aunque en su versión actual el *software* libre pero no *copyleft* seguirá estando disponible, siempre podrá ser mejorado y/o integrado en otro programa por parte de una empresa o proyecto, y estas mejoras se perderán si se publican con una licencia propietaria. Lo mismo ocurre con el *software* que está en el dominio público. En la gran mayoría de proyectos de *software* libre, y desde luego en el Proyecto Debian, son mucho más utilizadas las licencias *copyleft*, especialmente la GPL. Así, en la discusión que sigue hablaremos indistintamente de licencias libres o licencias *copyleft*. Aunque técnicamente no son equivalentes, evita la repetición y no altera el argumento. La existencia de *software* con licencias libres no *copyleft* no afecta a lo que se plantea sobre la constitución del procomún.

La insistencia que se produce en Debian y entre los defensores del *software* libre para no dejar el código en el dominio público, así como la preferencia que muchos tienen por las licencias *copyleft* frente a las que no lo son, muestran así una concepción del *software* como algo orgánico, que crece y mejora. Aunque siempre es posible tomar un programa que tenga una licencia libre y mejorarlo, volver a distribuirlo con las mejoras (cobrando por ello o no) supone la obligación de licenciar esas mejoras con una licencia también libre. No supone sólo la obligación de devolver lo recibido, sino también sus mejoras. Se trata de evitar que éstas se pierdan para la sociedad y el procomún (Debian.org, s.f.[u]). Aparece así también una concepción colectiva de la creatividad: si las mejoras creadas han de ser devueltas, es porque se apoyan necesariamente en la creación previa de otros individuos.

mons (Creativecommons.org, s.f.).

[17] El término en inglés, *release*, significa tanto «publicación» como «liberación».

[18] U ofrecer un esquema de licenciamiento dual junto a una licencia propietaria, como señalamos antes, aunque aquí no consideraremos ese caso.

Para la presente discusión es importante insistir en la idea de que las cuatro libertades (véase la página 23) que establece el *software* libre suponen siempre la libertad de seguir distribuyendo, donando, los objetos de que se trata, y que suponen el acceso al código fuente. Porque como hemos visto antes, el objeto del don en este caso no es simplemente el *software* que funciona en el ordenador, sino específicamente el código fuente.

Que podamos hablar de una obligación de devolver no significa entonces que al recibir un programa libre y usarlo haya que corresponder con la escritura de otro programa, sino que los cambios hechos a ese programa y su distribución se devolverán al procomún, al campo del *software* y la cultura libres. Está apareciendo entonces un sujeto nuevo, del que se toman y al que se devuelven los objetos. El del *software* libre es un don que no crea una deuda entre dos individuos particulares, sino en todo caso entre el procomún por un lado y quien quiera distribuir *software* basado en ese procomún. Podemos caracterizar al procomún como sujeto porque un Desarrollador se relaciona con él de un modo que no es reducible a una serie de intercambios individuales.

Esta obligación de devolver ha de ser matizada también en otro sentido. En las FAQ sobre las DFSG aparecen tres *tests* que ayudan a determinar si una determinada licencia es o no libre. Los dos primeros son el test de la isla desierta y el test del disidente (Pearlmutter, s.f., pregunta 9):

> a. **The Desert Island** test.
>
> Imagine a castaway on a desert island with a solar-powered computer. This would make it impossible to fulfill any requirement to make changes publicly available or to send patches to some particular place. This holds even if such requirements are only upon request, as the castaway might be able to receive messages but be unable to send them. To be free, software must be modifiable by this unfortunate castaway, who must also be able to legally share modifications with friends on the island.
>
> b. The **Dissident** test.

> Consider a dissident in a totalitarian state who wishes to share a modified bit of software with fellow dissidents, but does not wish to reveal the identity of the modifier, or directly reveal the modifications themselves, or even possession of the program, to the government. Any requirement for sending source modifications to anyone other than the recipient of the modified binary—in fact any forced distribution at all, beyond giving source to those who receive a copy of the binary—would put the dissident in danger. For Debian to consider software free it must not require any such "excess" distribution.

Es decir, en definitiva, la obligación se reduce a distribuir (devolver) las mejoras a aquellos a los que se distribuyen los binarios, y no puede aparecer la obligación explícita de proporcionárselas al autor original, ni siquiera a que sean accesibles al público en general (siempre que no se distribuyan los binarios). El procomún del que hablábamos hace un momento puede ser circunstancialmente reducido. Sin embargo, se mantiene el hecho de que distribuir *software* libre impide quitar derechos relativos al mismo a cualquiera que podría disfrutarlos una vez que los tiene. La cuestión es que no puede haber distribución de binarios sin que se constituya un procomún respecto al código fuente (y el conocimiento que encarnan) conformado al menos por aquellos a quienes se les distribuyen. Sobre esta cuestión trata justamente el tercer test, el de los «tentáculos del mal»:

> c. The **Tentacles of Evil** test.
>
> Imagine that the author is hired by a large evil corporation and, now in their thrall, attempts to do the worst to the users of the program: to make their lives miserable, to make them stop using the program, to expose them to legal liability, to make the program non-free, to discover their secrets, etc. The same can happen to a corporation bought out by a larger corporation bent on destroying free software in order to maintain its monopoly and extend its evil empire. The license cannot allow even the author to take away the required freedoms!

El título de esta tesis recoge esta transformación que suponen las licencias:

del *hau* a la GPL (u otras licencias) como el dispositivo que mantiene los bienes en circulación, ayudando a constituir el colectivo sociotécnico.[19] A diferencia del *hau*, la GPL es un artefacto cuya contribución a la estabilidad de los colectivos en los que se integra y las prácticas sociales que configura puede ser seguida y reconstruida empíricamente. Pero lo que conviene recoger y mantener de la antigua noción de *hau* es la sugerencia sobre la agencia de los objetos del don, como se ve en la conocida cita de Mauss:[20]

> A propósito del *hau*, del espíritu de las cosas [...] Tamati Ranaipiri [...] nos proporciona, por pura casualidad y sin prevención alguna, *la clave del problema* [...] si el regalo recibido e intercambiado obliga es porque la cosa recibida no es algo inerte. Aunque el donante la abandone, aún forma parte de él. A través de ella, se apodera del beneficiario [...]. En el fondo, *es el* hau *el que quiere volver al lugar de su nacimiento*, al santuario del bosque y del clan, y a su propietario [...] en el derecho maorí, el vínculo de derecho, un vínculo que se expresa a través de las cosas, es un vínculo del alma, pues la cosa misma tiene un alma [...]. Animada, a menudo individualizada [...] tiende a regresar a su «lugar de origen» o bien a producir, para el clan y la tierra de que forma parte, *un equivalente que la reemplace* [...] ((Mauss, 2009), citado por Godelier (1998, pág. 30)).

Una agencia que ya no depende de su radicación en un sujeto más o menos espiritual, sino del hecho de que las licencias son dispositivos que sólo funcionan por su asociación con otros elementos (el código fuente: normalmente la licencia es un archivo más del mismo; las diferentes legislaciones sobre *copyright*; la *Free Software Foundation* o *Software in the Public Interest*, organizaciones encargadas de defenderlas incluso en los tribunales, etc.) muy variados, reconfigurando

[19] Tiempo después de ponerle título a la presente tesis, descubrí la siguiente cita en un texto no publicado de Kelty: «If Free software licenses are the hau that compels the recipient to return something, and thus sediments its status as a particular kind of sacred property, then they are not alienable from the community that possesses them, i.e. the people who have decided to submit themselves to the use and development of Free software for whatever reason» (Kelty, 2002, pág. 73).

[20] También los objetos de cobre entre los haïda y los kwakiutl (Godelier, 1998, págs. 30-31). Sobre las propiedades de los objetos donados que las asimilan a personas véase Graeber (2001, págs. 166-167).

las cadenas de acciones y redistribuyendo la agencia a lo largo de éstas (véase la sección 4.2, *El código como actante*). Lo importante es no centrarse en la cosa, el objeto aislado, ni en las acciones o prácticas aisladas, sino en la red, las cadenas de prácticas, y sus restricciones. Esto es algo que vale tanto para entender el *software* libre como el *kula* o el *potlatch*. Tanto en Mauss como en la Teoría del Actor-Red, la agencia de los objetos proviene de su capacidad de establecer vínculos. No la tienen por sí mismos, lo que supondría olvidar precisamente que la agencia surge siempre de asociaciones, de relaciones. La capacidad de actuar depende de esos vínculos. Recordemos que ningún objeto la tiene por sí solo, pero un ser humano tampoco.

Y las licencias modifican la agencia como mediadores y no como intermediarios (véase la sección 4.2.2, *Mediadores e intermediarios. Las figuras del* bug *y del* hacker), introduciendo variaciones y cambios en el sentido de las acciones difícilmente previsibles. Las licencias son complejas como los programas y por la misma razón. Por eso pueden tener el equivalente a *bugs*. Esa es la razón de que, aunque legalmente sea posible escribir una licencia propia y específica, personal, esto es algo que se recomienda no hacer por sus consecuencias imprevisibles. Aparecen así las licencias casi-libres, licencias que en su diseño pretenden ser libres pero introducen alguna condición que impide que sea así, como «sólo se puede cobrar una pequeña cantidad por la redistribución» o «todos los arreglos de *bugs* deben enviarse al autor». Por eso las DFSG son guías generales, nunca un contrato o una serie de reglas cuyo seguimiento asegure el objetivo que se pretende. Han de ser interpretadas, y esta interpretación siempre es abierta (Pearlmutter, s.f.).

Este carácter de artefacto, de mediador, de objeto que da consistencia a las cadenas de acciones, hace de las licencias libres miembros de pleno derecho de los colectivos sociotécnicos donde se insertan y, como veremos enseguida, constituyentes esenciales del procomún digital. De Paoli, Teli *et al.* (2008) tratan de responder a la cuestión de cómo las licencias participan en la vida de las comunidades de *software* libre y abierto. Concluyen que:

> artefacts have to be considered in the study of FLOSS, because of their ability to connect different social worlds and to support socio–technical

> interactions. Looking at licenses as artefacts and at social practices, we have been able to show that they participate in the construction of both relational ecologies and political and technical boundaries. From this point of view, these legal artefacts act as boundary objects (Star and Griesemer, 1989), co–participating in the construction of an ecology of practices in community life. In our cases, conflicts and agreements relationally redefine legitimate participants, network of alliances, artefacts, technologies, and communities themselves (De Paoli, Teli *et al.* 2008).

También Callon señala el papel de los dispositivos legales en la configuración de la agencia a lo largo de los agenciamientos en los que se integran:

> Everything in these agencements that makes it possible to locate sources of action, establish origins, assign responsibilities, and account for profits and losses associated with a particular action, plays a strategic part in shaping agencies. In particular, I have in mind copyrights, property rights, human rights, etc. (Callon, 2005, pág. 4).

Las licencias libres serían así uno de los objetos híbridos por excelencia en el ámbito de esta investigación (el carácter híbrido de un determinado objeto es siempre relativo a una perspectiva de investigación), posiblemente incluso más que el *software* híbrido del que hemos hablado en el capítulo 4, *El carácter híbrido del* software. Bugs, hackers, *paquetes y repositorios*. De aquí el lugar central que ocupa el presente capítulo sobre las licencias y el don: determinados aspectos técnicos del desarrollo y la distribución de *software* llevan a la necesidad de determinadas licencias, que se convierten así en un dispositivo central y esencial. Y éstas a su vez nos van a conducir a la consideración de aspectos que podríamos llamar «sociales» como la significación de la «comunidad» o de los conceptos de «abierto» y «libre». Vamos a considerar ahora esta segunda cuestión.

5.2.2 Sobre el sentido de «libre» en el *software* libre y la cultura libre.

El *copyright* (o *copyleft*) y las licencias redistribuyen la agencia potenciando o disminuyendo el control que ejercen los desarrolladores y los usuarios sobre el *software* y sobre sus ordenadores. La *Free Software Foundation* define la «libertad» del *software* precisamente como control (Gnu.org, s.f.[c]):

> We campaign for these freedoms because everyone deserves them. With these freedoms, the users (both individually and collectively) control the program and what it does for them. When users don't control the program, we call it a "nonfree" or "proprietary" program. The nonfree program controls the users, and the developer controls the program; this makes the program an instrument of unjust power.

En la página *web* de donde procede la cita, la expresión «an instrument of unjust power» es un enlace a un texto de Stallman en el que se puede observar con claridad que la cuestión central cuando hablamos de la libertad del *software* es la capacidad de agencia, y no sólo en las actividades directamente relacionadas con la computación:

> With free software, the users control the program, both individually and collectively. So they control what their computers do (assuming those computers are loyal and do what the users' programs tell them to do).
>
> With proprietary software, the program controls the users, and some other entity (the developer or "owner") controls the program. So the proprietary program gives its developer power over its users. That is unjust in itself, and tempts the developer to mistreat the users in other ways.
>
> Freedom means having control over your own life. If you use a program to carry out activities in your life, your freedom depends on your having control over the program. You deserve to have control over the

> programs you use, and all the more so when you use them for something important in your life.
>
> [...]
>
> If the users don't control the program, the program controls the users. With proprietary software, there is always some entity, the developer or "owner" of the program, that controls the program—and through it, exercises power over its users. A nonfree program is a yoke, an instrument of unjust power (Stallman, 2015).

Así, *software* libre es el que proporciona al usuario control sobre el mismo. En términos de la Teoría del Actor-Red, esto quiere decir que el usuario puede establecer nuevas traducciones y asociaciones en su relación con el *software*, y en las que establece a través suyo. La libertad del *software* se puede entender como una manera de permitir asociaciones entre el *software* y los usuarios que dan más agencia a éstos que al propietario. En los términos de la semiótica de Kockelman, que ofrecen un grado mayor de agencia residencial (véase la sección 2.2.8, *Agencia y semiótica*). Esta redistribución de la agencia es posible porque las cajas negras en las que consiste el *software* se caracterizan, en este ámbito, porque su apertura y modificación es siempre posible (véase la sección 2.2.6, *De las cajas negras a los parches como metáfora de lo técnico*). Para esto, las licencias libres son esenciales y por ello pueden considerarse, como lo han sido en numerosas ocasiones, un *hack* legal (en tanto que subvierten las normas sobre el *copyright*, utilizando esas mismas normas). En esta redistribución, como ponen de manifiesto las citas de la *Free Software Foundation* y de Stallman, no sólo se constituye un orden tecnológico diferente de acción, sino que al mismo tiempo se trata de un orden legal y un orden ético. Lo que tratamos de comprender en este trabajo es cómo configura al colectivo considerado, cómo lo estabiliza, el hecho de que sus objetos sean dones, lo que en este caso quiere decir ni más ni menos que tienen una licencia de determinado tipo. Ésta es una cuestión central: las licencias son el objeto que constituyen el *software* o el código como don, en una operación que al mismo tiempo determina que lo constituyan también simultáneamente como libre. Don y libertad son aquí inseparables, y ambas dimensiones se articulan en la licencia. De ahí la utilización del término «libre», con todas sus connotaciones, para definir un agenciamiento técnico-legal. Las licencias y el diseño de la infraes-

tructura sólo consiguen movilizar a los agentes si se movilizan también al mismo tiempo el deseo y la imaginación de los participantes.

Entonces, el uso del término «libre» en los conceptos de «cultura libre» y «*software* libre» no supone un ideal abstracto, sino un conjunto preciso de derechos de uso que se conceden al receptor de esas obras (modificando por tanto su capacidad de agencia), y que se especifican en las licencias usadas. Los discursos y prácticas en torno a este ámbito se refieren a condiciones concretas de actividad en las que lo *libre* ha sido claramente articulado. Por esto se trata la cuestión del significado de términos tan importantes para este ámbito como «libre» o «abierto» en este capítulo sobre el don, porque la libertad es aquí algo bien específico y definido en relación a las licencias y a la agencia que permiten, y como veremos enseguida, al procomún. Se introduce así también un campo de significación nuevo respecto a lo que se ha llamado economías tradicionales del don.

Podríamos aplicar al *software* libre las siguientes consideraciones de David Graeber respecto a la relación entre la libertad y la capacidad de establecer vínculos:

> The meaning of the Roman word *libertas* itself changed dramatically over time. As everywhere in the ancient world, to be «free» meant, first and foremost, not to be a slave. Since slavery means above all the annihilation of social ties and the ability to form them, freedom meant the capacity to make and maintain moral commitments to others. The English word «free,» for instance, is derived from a German root meaning «friend,» since to be free meant to be able to make friends, to keep promises, to live within a community of equals. This is why freed slaves in Rome became citizens: to be free, by definition, meant to be anchored in a civic community, with all the rights and responsibilities that this entailed (Graeber, 2011, pág. 203).

Por otra parte, Latour estaría de acuerdo con la inseparabilidad de elementos legales, éticos y tecnológicos en estos agenciamientos:

> We have been able to delegate to nonhumans not only force as we have known it for centuries but also values, duties, and ethics. It is because of this morality that we, humans, behave so ethically, no matter how weak and wicked we feel we are. The sum of morality does not only remain stable but increases enormously with the population of nonhumans. It is at this time, funnily enough, that moralists who focus on isolated socialized humans despair of us –us meaning of course humans and their retinue of nonhumans (Latour, 1992, pág. 232).

Y aquí se manifiesta la importancia de la reciprocidad, la cooperación y la solidaridad como componentes de las relaciones basadas en el don. No como una estructura primordial como ocurría en Lévi-Strauss, sino como algo producido y articulado por diferentes mecanismos. El mecanismo descrito que pretende mantener los objetos en circulación es entendido como sostén de las relaciones sociales en tanto supone una condición de posibilidad de la cooperación. Así lo expresa Stallman:

> Freedom includes the freedom to cooperate with others. Denying people that freedom means keeping them divided, which is the start of a scheme to oppress them. In the free software community, we are very much aware of the importance of the freedom to cooperate because our work consists of organized cooperation. If your friend comes to visit and sees you use a program, she might ask for a copy. A program which stops you from redistributing it, or says you're "not supposed to", is antisocial (Stallman, 2015).

Volvemos a encontrar aquí la doble articulación de orden tecnológico y orden ético y social que vimos en la sección 3.3, *Debian como público recursivo*. Darle significado en las prácticas sociales a las ideas de reciprocidad, cooperación o solidaridad requiere darles un cuerpo técnico y legal. ¿Cuál de estas dos dimensiones tiene prioridad explicativa? Si queremos entender el fenómeno del *software* libre, ¿es más acertado centrarse en los modos en que los sujetos sociales imaginan sus prácticas, o en las leyes y mecanismos objetivados que las regulan?

Cuando discutimos la libertad del *software* o la reciprocidad que supone, ¿nos referimos a las disposiciones y motivaciones presentes en las representaciones de los sujetos respecto a su acción, o al funcionamiento de los mecanismos y dispositivos que regulan su producción y distribución?

O expresado de otra manera: ¿lo que constituye el colectivo/comunidad son las ideas y representaciones que comparten sus integrantes, o más bien las reglas técnicas y legales a las que se someten? Por supuesto, ambas cosas no se pueden separar. Más que de prioridad explicativa, se trata entonces de seguir la articulación concreta de estas dimensiones. Lo seguiremos viendo en el capítulo siguiente a propósito de la comunidad y/o colectivo del *software* libre y Debian. Ahora intentaremos ilustrar esta cuestión a propósito de una controversia persistente sobre el significado del *software* libre y sus licencias, la que se da entre los términos *software* libre (*free software*) y *software* de código abierto (*open source software*).

Podemos distinguir tres grandes perspectivas sobre el significado del *software* libre, con diferentes implicaciones morales y sociales. Además de la de Debian expuesta en las *Debian Free Software Guidelines*, las dos más influyentes son las de la *Free Software Foundation*, que usa la expresión «*software* libre» y la de la *Open Source Initiative*, que prefiere «*software* de código abierto».[21] La definición de *software* libre de la FSF depende de las cuatro libertades ya citadas (véase la página 23, también en Gnu.org (s.f.[c])), mientras que la definición de *open source software* de la OSI se concreta en «The Open Source Definition» (Opensource.org, s.f.). Ésta se basa por su parte en la especificación del *software* libre de Debian, las DFSG, y ambas fueron en gran parte propuestas por un anterior DPL de Debian, Bruce Perens. En esta controversia, Debian ocupa de hecho una posición un tanto paradójica, en tanto que, por una parte, el compromiso con los valores que se presentan en el manifiesto y el contrato social acercan su posición a la de la FSF y, por otra parte, las DFSG son la base de la *Open Source Definition*.[22] En cualquier caso, el término preferido y usado en Debian es el de

[21] También se utiliza, evitando esta controversia, la expresión FLOSS o F/LOSS, *Free/Libre Open Open Source Software*. La expresión Free/Libre (tal como se usa en inglés) ayuda además a eliminar una fuente común de confusión, el doble sentido del término inglés *free* como libre y como gratis.

[22] Esta cierta ambigüedad se observa también en el mismo Perens, que sin dejar el uso de *open source*

software libre (Debian.org, s.f.[u]; Pearlmutter, s.f.).

La cuestión es que esta controversia[23] se refiere a diferencias filosóficas y a distintas concepciones morales y sociales, pero las diferentes definiciones se aplican a las mismas licencias. Si una licencia es libre de acuerdo a una de las tres definiciones, lo será también de acuerdo con las otras dos. Extensionalmente, *software* libre y *software* de código abierto son equivalentes.[24] Volviendo a la consideración semiótica de la agencia que vimos en la sección 2.2.8, *Agencia y semiótica*, podríamos decir que «libre» y «abierto» son dos signos que constituyen dos objetos diferentes, no porque distingan dos tipos de *software* o de licencias, sino porque constituyen dos formas diferentes de comunidad o de colectivo, proponiendo diferentes interpretantes. La motivación fundamental de la OSI fue hacer aceptables este tipo de licencias en un entorno empresarial, por lo que enfatizaron cuestiones relativas a las ventajas pragmáticas del código (fiabilidad, calidad, coste, etc.) más que la capacidad de control propio que podía proporcionar a los agentes ajenos a las empresas.[25] Por el contrario, hablar de *software* libre supone poner de manifiesto las implicaciones éticas y políticas de la participación en la comunidad (Gnu.org, s.f.[d]). Como vimos en la sección 2.2.8, *Agencia y semiótica*, la expresión de estos signos y la creación de interpretantes contribuye a determinar la agencia residencial de los agentes implicados. En concreto, el uso del término «libre» en lugar de «código abierto» contribuye a crear el colectivo formado en torno a los vínculos técnicos que estamos viendo como una comunidad constituida también moral y socialmente en torno a determinados principios.

Referida a Debian como Proyecto, más allá de las obligaciones de sus Desarrolladores individuales y yendo más allá de las obligaciones legales impuestas por las licencias, la obligación de devolver aparece en el segundo punto de su Contrato Social (véase el apéndice A, *Contrato social*), donde se establece que:

Contribuiremos a la comunidad de *software* libre.

también insiste en la importancia de la libertad. Véase Perens (1999).

[23] Se puede seguir un desarrollo histórico de la misma en Kelty (2008, cap. 3). También se puede ver la pregunta 31 de las «DFSG and Software License FAQ», en Pearlmutter (s.f.).

[24] Tal como se establece por ejemplo en Debian.org (s.f.[u]).

[25] Sobre esto puede verse también Hill (2009).

> Cuando escribamos nuevos componentes del sistema Debian, los licenciaremos de forma consistente con nuestra definición de *software* libre. Haremos el mejor sistema que podamos, de forma que el *software* libre tenga amplia difusión y uso. Enviaremos parches, mejoras, peticiones de los usuarios, etc. a los autores originales (esto se conoce en inglés como «upstream», N. del T.) del *software* incluido en nuestro sistema.

5.2.3 Las licencias en Debian.

Pero, ¿cómo se integran las licencias en la vida cotidiana y las prácticas sociales del Proyecto Debian? ¿En qué agenciamientos toman parte? ¿Cómo redistribuyen la agencia? Como vimos en el capítulo anterior, lo que Debian como distribución y sus miembros donan específicamente son fundamentalmente paquetes de *software*. No tanto el código fuente, que usualmente ya está donado por *upstream*, sino su empaquetado, es decir la posibilidad de integrarlo en un sistema. También hay código fuente desarrollado en Debian, pero lo cierto es que la mayoría de sus prácticas se refieren al empaquetado de *software* preexistente.

Además, recordemos que los paquetes más importantes eran los paquetes fuente. En éstos, una de las cuestiones de las que tiene que asegurarse el mantenedor es de que tiene una licencia adecuada, y que toda la información relevante sobre la misma está incluida. El archivo que la recoge dentro del paquete es `debian/copyright`, que debe quedar instalado en el ordenador donde se instala el paquete. El «Manual de Normas de Debian» (*debian-policy*)[26] establece que en este archivo debe aparecer la licencia completa (o, en el caso de licencias ampliamente utilizadas como GPL o BSD, enlazar a un documento que la contenga).

Aunque es opcional y no obligatorio, se recomienda que este archivo siga un determinado formato[27] que permita que sea leído e interpretado tanto por humanos como por ordenadores, estandarizando la forma en que se presenta la información de tal modo que se pueda extraer automáticamente información sobre las

[26] El documento que detalla la estructura y contenido del archivo, el diseño del sistema operativo y las normas que debe cumplir un paquete para ser incluido en la distribución. Puede consultarse en Debian.org/doc (s.f.[d]).

[27] Véase «Machine-readable `debian/copyright` file», en Debian.org/doc (s.f.[h]). El origen de este documento es una *Debian Enhancement Proposal* (DEP) accesible en Dep-team (s.f.[e]).

licencias y los requerimientos legales de los diferentes paquetes del archivo. Esto posibilita la integración técnica de la información legal en la infraestructura de la distribución, así como la comprobación automatizada de la compatibilidad de las diferentes licencias. Un uso extendido de este formato contribuirá a mantener el archivo de Debian libre y solucionar buena parte de las posibles incompatibilidades.

El archivo de Debian no contiene exclusivamente *software* libre. Se pueden distinguir tres áreas en el archivo: *main*, que forma propiamente la distribución Debian y sólo contiene *software* libre según la definición de las *Debian Free Software Guidelines*; *non-free*, que contiene *software* no libre según las DFSG pero que se puede distribuir legalmente; y *contrib*, que contiene *software* que aun siendo libre en el sentido especificado necesita que se instale algún *software* no libre para funcionar. El *software* de las dos últimas secciones no es propiamente parte de la distribución, y no lo tendremos en cuenta en lo que sigue (Debian.org/doc, s.f.[d], cap. 2).

La responsabilidad de que todos los paquetes subidos a Debian sean efectivamente *software* libre y legalmente redistribuibles es de cada Desarrollador o Mantenedor individual. Éste ha de asegurarse de que la licencia usada por *upstream* cumple con las DFSG. Hay una lista de correo donde se pueden plantear públicamente, entre otras cosas, dudas sobre la corrección de una licencia o su compatibilidad con Debian, `debian-legal` (Debian-legal, s.f.). Esta lista no representa la posición oficial de Debian sobre algunas licencias, es meramente una lista para consultas. La primera vez que un determinado paquete se sube al archivo entra en la llamada «NEW queue», [28] donde el FTP Team comprueba que el paquete puede efectivamente formar parte de la distribución, y lo rechaza si no es así. Hay comprobaciones técnicas de todo tipo antes de que un nuevo paquete se integre en la distribución, pero la más importante es precisamente la comprobación de la licencia, que determina si el paquete es legalmente redistribuible. Esta comprobación es manual y está a cargo de los miembros del equipo, lo que hace que tengan que dominar las complejidades de las cuestiones legales implicadas. Si el paquete es una actualización de un paquete ya presente en el archivo, no es ne-

[28] Véase Wiki.debian.org (s.f.[ai]). Se puede comprobar qué paquetes están allí en un determinado momento en Ftp-master.debian.org (s.f.[b]).

cesaria esta comprobación y se sube directamente al archivo sin pasar por NEW. Una cuestión interesante es que la decisión se toma relativa a cada paquete, no a la licencia. Aun si un paquete tienen una determinada licencia, es necesario comprobar que ese paquete en concreto no tiene algún otro problema que impida su redistribución.[29]

Para discutir algunos elementos de este proceso en relación con las cuestiones que estamos tratando aquí y en otras partes de esta investigación, me referiré al caso narrado por Bradley Kuhn, miembro destacado de la *Free Software Foundation* y de la organización (véase también más adelante) *Software Freedom Conservancy*, en una conferencia titulada «The Supreme Court of DFSG-Free» ofrecida en DebConf 16, una de las conferencias anuales de Debian.[30] Kuhn es el inventor de la licencia Affero, Affero GPL o AGPL, derivada de la licencia GPL pero con una cláusula que extiende el *copyleft* al *software* usado para ofrecer servicios en red, y que como tal no es necesario distribuir a los usuarios para que lo usen. En dicha conferencia, Kuhn cuenta como el estatus de esta licencia como licencia libre o no era una cuestión disputada antes del año 2008. En un determinado momento la *Open Source Initiative* la incluyó en su listado de licencias libres, pero la decisión no era definitiva. Esta indecisión, junto con el hecho de que grandes corporaciones (y por razones obvias) como Google prefiriesen que dicha licencia se considerase no libre, hacía temer que el consenso general podía decantarse por su no admisión como licencia libre. La OSI y la FSF son probablemente las dos instancias más importantes para determinar si una licencia es libre o no, pero según Kuhn la tercera es, y debe ser reconocida como tal por buenas razones, precisamente Debian.

En este caso Debian podía entonces contribuir a resolver definitivamente la cuestión. Sin embargo, Debian no tiene un procedimiento establecido para determinar si una licencia es o no libre. Están las DFSG, pero como hemos visto éstas no son un contrato o un procedimiento inequívoco, sino que han de ser interpretadas. Y la lista `debian-legal` es meramente consultiva, sin poder de decisión. Donde recae el verdadero poder de decisión es en el FTP Team. Si éste acepta un paquete

[29] Véase Debian.org (s.f.[v]). Aquí puede verse una lista de licencias generalmente admitidas en Debian.

[30] El video se puede ver en Debconf16.debconf.org (s.f.[c]). Lo que sigue recorre buena parte de la argumentación de dicha charla y depende de ella.

subido a NEW con una nueva licencia, esto es de hecho un fuerte argumento a favor de su aceptación general como licencia libre. Así que la estrategia de Kuhn consistió en conseguir que se aceptara un paquete (en este caso `yocto-reader`) licenciado con la AGPL en el archivo de Debian, que subió un Desarrollador, forzando así al FTP Team (y con él, de alguna manera, a Debian) a decidir. De hecho, la cuestión había sido discutida previamente en la lista `debian-legal` y el consenso (en ningún modo vinculante) parecía ser que la AGPL no debía ser considerada libre. El mismo Kuhn señala que fue un ingenuo al pensar que la opinión surgida de `debian-legal` tenía alguna autoridad. Pero una vez que el paquete fue aceptado en el archivo y pasó a formar parte de la distribución, se estableció por la vía de los hechos que Debian aceptaba dicha licencia como licencia libre. Y con Debian, gran parte de la más amplia comunidad del *software* libre. Es cierto que, como hemos visto y como un miembro del equipo le confirmó a Kuhn, el FTP Team no decide sobre licencias, sino sobre paquetes concretos. Sin embargo, en esta ocasión (sólo en ésta) sí que se hizo una declaración general en relación a la AGPL, comentando los argumentos que se habían tenido en cuenta.[31]

La cuestión que se plantea Kuhn es por qué es el FTP Team quien decide sobre estas cuestiones de facto, incluso aunque la interpretación de las DFSG corresponde en principio al Proyecto en su conjunto. Lo que le preocupa es la opacidad del proceso, llegando a proponer una especie de «corte de apelación» o «corte suprema» que tome este tipo de decisiones, a semejanza del comité técnico (véase el cuadro 2.2, *Algunos poderes del Comité Técnico*). Sin embargo, lo importante de esta historia es que pone de manifiesto hasta qué punto la doocracia es una forma de tomar decisiones fundamental en Debian, y cómo supone una capacidad real de decidir. En primer lugar, hay que insistir en que se consiguió una toma de posición por parte de Debian cuando un Desarrollador que podía hacerlo subió un paquete, no a través de una discusión general que buscase el consenso. En segundo lugar, la decisión fue tomada por quien podía de hecho hacerlo y ponerla en práctica (esto no impide que la decisión tomada pudiera haberse rechazado a través de una Resolución General). Lo que determina el punto hasta el que uno puede decidir no depende de que se haya especificado ese ámbito intencionalmente en su diseño (como señala Kuhn, entre las atribuciones delegadas por el DPL al FTP Team no está explícitamente la de interpretar las DFSG), sino del control real sobre los

[31] Véase Debian-legal (2008). También Bugs.debian.org (s.f.[h]).

procesos.

Por otra parte, afirma también Kuhn que las licencias son documentos de naturaleza política (en tanto que afectan al comportamiento de otros), y que son de alguna manera las constituciones de las comunidades del *software* libre. De ahí que las DFSG sean un documento tan fundamental, no solo para Debian sino para la comunidad del *software* libre en general. Y que los FTP Masters ocupen de hecho puestos «políticos». Estas ideas coinciden con lo que venimos defendiendo en el presente trabajo. Pero uno de los supuestos para proponer una «corte suprema» para lo referente a las licencias es la separación de estas decisiones de aquellas que son «técnicas». La cuestión es que en la práctica ésta es una distinción que, como vimos en la sección 4.4, *El código como objeto híbrido*, se ha de producir a través de un trabajo de separación y purificación. Las licencias son objetos híbridos en tanto que están implicadas en ambas dimensiones. Eso es lo que hace que tengan un lugar tan constitutivo en los agenciamientos y cadenas de prácticas que se producen en el Proyecto Debian.

Así pues, para concluir, las licencias son los dispositivos que en mayor medida determinan si un determinado elemento de *software* llega o no a formar parte de la distribución. Un Desarrollador o Mantenedor necesita entonces un cierto conocimiento no sólo de cuestiones técnicas para mantener un paquete, sino también legal. Pero como hemos visto en la sección anterior, en la elección y uso de unas licencias en lugar de otras no están implicadas únicamente cuestiones de técnica legal, sino que encontramos también una dimensión que tiene que ver con los objetivos y significado últimos del Proyecto, la defensa del *software* que pueda ser considerado libre y, como seguiremos viendo, la constitución de una comunidad en torno a él. De ahí que el vínculo del *software* con la licencia que regula su vida sea uno de los vínculos centrales y determinantes en la red sociotécnica que intentamos describir.

Nos hemos centrado en las licencias que regulan la publicación y distribución del *software*. Pero hay más puntos en los que la distribución, creación o uso de productos tienen consecuencias legales, que podrían ser discutidas en el contexto del *software* libre como don. Fundamentalmente son la cuestión de las marcas registradas y de las patentes. Ambas cuestiones afectan al Proyecto Debian,

pero no modifican los argumentos que se querían exponer en este capítulo, por lo que aquí solamente se señalarán algunas referencias para quien esté interesado en la cuestión. Se puede encontrar una guía en una comunicación dada por el antiguo DPL Stefano Zacchiroli.[32] Sobre la cuestión de las marcas registradas (básicamente el logo oficial y el nombre de Debian) pueden consultarse «Debian Trademarks» (Debian.org, s.f.[r]) y «Discover Debian's hassle-free trademarks, use them to promote Debian» (Debian.org/News, 2013), y sobre las patentes de *software* que pueden afectar al proyecto, «Debian Position on Software Patents» (Debian.org, s.f.[w]) y «Community Distribution Patent Policy FAQ» (Debian.org, s.f.[x]), todas ellas en la *web* del Proyecto.

5.2.4 Procomún y cultura libre.

Pensar el *software* libre como don requiere situarlo también en el marco más general del procomún y lo que ha venido en llamarse *Cultura Libre* (Lessig, 2005). Ambos conceptos, el de don (o economías del don) y el de procomún, son independientes y no tienen por qué ir unidos, aunque ambos comparten el contraste con el mercado y la producción de mercancías. Pero en el caso del *software* libre sí encontramos que esas dos dimensiones son inseparables. Como mínimo, la retórica del don acompaña la constitución de este procomún.

La relación entre ambos conceptos, el más amplio de cultura libre o procomún digital y el más estrecho de *software* libre, es compleja. El segundo sería parte del primero (en tanto que el *software* es una porción de los productos culturales que pueden ser libres), pero la extensión de lo que ahora llamamos cultura libre sólo ha sido posible como una extrapolación a partir del caso de éxito que ha supuesto el *software* libre. Por otra parte, el *software* libre no sólo ha proporcionado un modelo extensible a otros productos culturales, sino que también funciona en muchas ocasiones como su infraestructura técnica, al menos de los bienes comunes digitales. En realidad, no he encontrado en el Proyecto Debian usos significativos del concepto de *commons* o procomún, aunque sí numerosas referencias al ámbito más amplio de la comunidad del *software* libre, empezando por el hecho de que su contrato social es un «contrato con la comunidad del *software* libre» (véase el apéndice A, *Contrato social*).

[32] En Zacchiroli (2013) y comentada en un artículo de Kerrisk (2013).

Moglen (1999), reconocido activista del movimiento del *software* libre, director del *Software Freedom Law Center* y consejero de la *Free Software Foundation*, establece claramente la creación de un procomún a partir del funcionamiento de la GPL:

> Section 2(b) of the GPL is sometimes called «restrictive,» but its intention is liberating. It creates a commons, to which anyone may add but from which no one may subtract. Because of §2(b), each contributor to a GPL'd project is assured that she, and all other users, will be able to run, modify and redistribute the program indefinitely, that source code will always be available, and that, unlike commercial software, its longevity cannot be limited by the contingencies of the marketplace or the decisions of future developers. [...]
>
> Users of GPL'd code, including those who purchase software and systems from a commercial reseller, know that future improvements and repairs will be accessible from the commons [...]
>
> Free software, at the risk of repetition, is a commons (Moglen, 1999).

He venido hablando, y lo seguiré haciendo, del procomún como si fuera una única entidad, singular. Este uso no presupone que no se puedan diferenciar diferentes ámbitos de procomún, o incluso diferentes instancias; o si, por el contrario, todo procomún forma un continuo. En todo caso, queda claro que nos referimos al procomún digital que se construye en torno al *software* libre y sus extensiones o modulaciones (Kelty, 2008) en la cultura libre.

Podemos observar una afinidad entre los conceptos de procomún y de público recursivo (véase la sección 3.3, *Debian como público recursivo*): habrá tanto más procomún en tanto que lo que es de todos no sean sólo las obras y los productos, sino también la gestión y construcción de los repositorios en los que se implementa el procomún. Podemos entender en general por procomún un repositorio común, a libre disposición de quien quiera usarlo, de recursos de diferentes tipos. En este caso, fundamentalmente programas de *software* libre.

El concepto de *procomún* (Benkler, 2006; Boyle, 2008; Ghosh, 2005a; Stalder, 2010) se ha mostrado especialmente útil para pensar las articulaciones entre dones y comunidad. Clippinger y Bollier (2005) han ofrecido una definición de procomún (*commons* en inglés) que pone de relieve algunas características interesantes:

> The commons is a social regime for managing shared resources and forging a community of shared values and purpose [...] In a commons, transactions are based on ongoing moral, social, and personal relationships, not episodic, impersonal exchanges of money. A commons is also marked by openness (Clippinger y Bollier, 2005, pág. 263).

Si el procomún es un conjunto de recursos compartidos que se poseen colectivamente, no se puede pensar si no es en referencia a una *comunidad* o colectivo. Además, y como esta definición pone de manifiesto, el procomún es algo más que los bienes que son compartidos: supone todo un régimen de relaciones que van más allá de las de propiedad, incluyendo relaciones morales y sociales, relaciones que incluyen los dispositivos que forman parte de ese procomún. Las cuatro libertades que constituyen la definición del *software* libre (véase la página 23) hacen referencia explícita a la importancia de la comunidad, sobre todo las dos últimas: «The freedom to redistribute copies so you can help your neighbor (freedom 2). The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this» (Gnu.org, s.f.[c]). De nuevo vemos que es una comunidad o colectivo que se articula también como público recursivo.

F. Turner (2009, pág. 76) establece dos condiciones para la producción colaborativa basada en los bienes comunes: la primera es la existencia de un procomún, un espacio en el que los miembros de mundos sociales diferentes se puedan reunir y colaborar, y que de alguna manera obtengan recursos económicos. Además,

> Alongside a commons and sufficient subsidy, online commons-based

> peer production depends on the interaction of some sort of communal ethos and multiple, non-monetary forms of compensation. [...] To make information goods valuable is to give 'gifts' to the 'community'. This shift in rhetorical frame from factory and market to gift and community in turn legitimates the multiple systems of reward actually in play [...] At the same time, rhetorics of mission and community allow collaborators to imagine that all participants, regardless of their actual standing, are in fact social and ethical peers (F. Turner, 2009, pág. 77).

La visión de Turner es crítica, en tanto nos recuerda que el componente retórico del procomún y la producción colaborativa es compatible con la desigualdad o la producción por el beneficio (como estamos viendo en este capítulo), pero indica también la importancia de la idea de procomún como un requisito para la producción colaborativa, además de su relación con la retórica de los dones y la comunidad, con un determinado imaginario ético y social.

En las dos últimas citas aparece también una contraposición clave entre transacciones impersonales basadas en el intercambio de dinero y el mercado, e intercambios basados en relaciones concretas y personalizadas: en definitiva, entre dones y mercancías. La diferencia entre estas dos formas ha sido descrita minuciosamente por Gregory (1982, 1997). Aquí destacaremos una de las diferencias fundamentales que establece, la inalienabilidad de los dones:

> A logical opposition between gifts (relations between non-aliens by means of inalienable things) and commodities (relations between aliens by means of alienable things) is the primary distinction (Gregory, 1997, pág. 53).

Puesto que la cosa donada no es alienada, se mantienen algunos derechos sobre ella. Ya vimos como lo expresaba Godelier a propósito del intercambio de dones entre los baruya. Por su parte, Weiner introduce un matiz en esa distinción que le permitirá introducir el concepto de «posesiones inalienables». Como los dones (éstos, como veremos, sólo se pueden entender en relación a aquéllas) y a diferencia de las mercancías, mantienen una vinculación íntima con los donantes:

> Some things, like most commodities, are easy to give. But there are other possessions that are imbued with the intrinsic and ineffable identities of their owners which are not easy to give away. Ideally, these inalienable possessions are kept by their owners from one generation to the next within the closed context of family, descent group, or dynasty. The loss of such an inalienable possession diminishes the self and by extension, the group to which the person belongs (Weiner, 1992, pág. 6).

La extrañeza que produce el hecho mismo del don en general, y especialmente la idea de que el objeto retiene algo de la identidad del donante, proviene de que nos hemos acostumbrado a considerar las cosas intercambiadas como mercancías, separadas e independizadas de su productor. Sin embargo, en el caso del *software* libre el vínculo entre quien dona y lo donado viene determinado por la licencia utilizada. Ahora bien, para comprender adecuadamente esto es preciso hacerlo teniendo en cuenta el papel que juega el procomún en este proceso.

Porque la licencia libre no establece directamente sólo un vínculo entre el autor individual y el programa, sino también entre éste y el procomún, un vínculo de inalienabilidad. Un objeto licenciado con una de estas licencias no se dona directamente a un individuo, ni siquiera a una clase de individuos, sino que establecen una cesión al procomún. El vínculo que liga a la cosa con su autor se produce a través de la asignación de *copyright*, pero los derechos de uso la ligan al procomún. Son las licencias libres las que constituyen el entramado del procomún. Es cierto que la posesión individual del *copyright* se retiene por parte del autor original del *software* incluso cuando lo dona. De hecho, el autor siempre puede hacer una obra derivada privativa, retirándola así del procomún (Pearlmutter, s.f., pregunta 29):

> Q: If I release software under a free software license that does not allow others to make proprietary derived works, does this preclude me from making proprietary derived works myself?
>
> A: No.

> Licenses gives others permissions that you already have. It is your code. You don't need your own permission to make a proprietary version, or to release a version under a different license, or to sell someone the right to make a proprietary version, or to sell someone the right to incorporate parts of your code in a proprietary program. (One caveat: if you incorporate non-trivial changes other people have made into your code base you are no longer the sole author. You would then need their permission to make a proprietary version, just as they would need yours.)

El donante original, el autor del código, mantiene la propiedad inalienable del objeto. Según Godelier (1998, págs. 82-83), aquí estaría una de las principales aportaciones de Sahlins sobre el don:

> a saber, que el donante original no deja de tener derechos sobre el objeto que ha donado, sea cual fuere el número de personas entre las que circule dicho objeto. Por supuesto, el hecho de que circule significa que todos los que lo reciben, todos sus donatarios, se convierten a su vez en donantes. Sin embargo, ninguno de ellos tendrá jamás los mismos derechos sobre el objeto que el primero de los donantes. Éste conserva la propiedad inalienable del objeto, mientras que los demás gozan de ciertos derechos de posesión y de uso del objeto que son alienables y temporales, y que se transfieren de uno a otro.
>
> El hecho esencial de la permanencia de los derechos del donante inicial sobre la cosa donada es el que se traduce en el plano ideal (dicho de otro modo, el de las representaciones-interpretaciones indígenas de dicha permanencia) mediante la idea de que la persona del donante inicial se halla presente en la cosa donada, de que queda atada a ella y la acompaña luego en todos esos cambios de mano y de lugar.

Tenemos así una clara distinción entre el donante original y los demás sujetos que toman parte en la circulación, por la que el primero mantiene, en nuestro caso, los derechos morales y el *copyright*, mientras que los demás disfrutan de

un derecho de uso libre. Una distinción que permite según Godelier mantener la discusión en el ámbito de los derechos de propiedad sin necesidad de usar las representaciones indígenas.

En cualquier caso, lo que el autor original sigue sin poder hacer es retirar la versión ya liberada o publicada, como veíamos a propósito del test de «los tentáculos del mal». Esto nos permite establecer que efectivamente se da una relación de donación especial entre el donante original y el procomún. Además, si otros autores han contribuido significativamente necesita su permiso para hacer una versión propietaria, lo que dificulta enormemente que esto ocurra en proyectos grandes de *software* libre.

Recordemos el hecho ya mencionado de que lo que las licencias prohíben es la apropiación privada de los bienes comunes y leamos el siguiente texto de Godelier:

> [...] nos enfrentamos precisamente a eso que tanto había fascinado a Mauss: que haya en la cosa donada una «fuerza» que actúa sobre quien la recibe y lo obliga a «devolverla». Hemos observado que esta fuerza reside en el hecho de que la cosa o la persona no resultan alienadas cuando se dona. Por el contrario, la cosa donada sigue formando parte de las realidades que conforman la identidad, el ser, la esencia inalienable de un grupo humano, de una persona moral. Podríamos decir que se trata de un «bien» común del que puede cederse el uso, pero nunca la propiedad. La cuestión que se plantea entonces es saber cuáles son las razones de la inalienabilidad de ciertas realidades que son propiedad de los grupos humanos, ya se trate de la tierra, de los objetos sagrados, de las fórmulas rituales, etc. (Godelier, 1998, pág. 70).

Como los dones que circulan en el *kula* (Godelier, 1998, págs. 134-141), los dones del *software* libre están sujetos a la vez a los principios de alienabilidad («a condición de que el objeto poseído no salga del marco del *kula*») e inalienabilidad. Al ser bienes digitales, es fácil comprender como pueden circular y no circular a la vez, en dos sentidos diferentes. Son inalienables y excluidos de la circulación en

tanto que siguen formando parte del procomún y no son apropiables directamente; y son alienados por su creador y circulan en tanto que son cedidos al procomún y cualquiera los puede tomar de ahí y redistribuirlos.

Así, aunque el concepto de procomún en tanto forma específica de propiedad tiene una larga historia, lo cierto es que su actualidad y, sobre todo, su desarrollo están ligados a los bienes y procesos susceptibles de digitalización y distribución a través de redes digitales. Lo que aparece como novedad sobre todo es la idea de procomún digital, compuesto por bienes que pueden ser digitalizados pero sobre todo *licenciados*. Se evitan así los problemas que se refieren a la llamada «tragedia de los comunes» (su no sostenibilidad a largo plazo debido al uso egoísta y exclusivo de los bienes comunes) popularizada por Hardin (1968). Boyle (2005, 241 ss.) señala algunas diferencias que hacen que el ámbito del procomún digital no pueda ser equiparado al tradicional. Las más importantes son la infinita copiabilidad y los costes de distribución tendentes a cero. Son estas características las que según Kollock (1999, pág. 223) facilitan los dones en entornos *online*.

Nadie puede por lo tanto quedar excluido de los bienes comunes que forman parte del procomún del *software* libre. Lo inalienable se refiere al grupo, a la relación con el grupo. Lo que pretende hacer Weiner (1992) es precisamente explicar las reglas del don a partir de la paradoja que supone el «guardar para (poder) donar». Como dice Weiner (1992, pág. x), «[s]earching for the kinds of possessions that people try to keep out of circulation is far more theoretically meaningful than assuming that exchange simply involves the reciprocity of gift giving». Y como hemos visto, en el caso del *software* libre lo que no se da, lo que se mantiene fuera de la circulación, es lo que se da a la comunidad, al procomún.

Dos Desarrolladores expresaban así la idea de la propiedad social, colectiva, de aquello que se dona y en lo que se fundamenta el procomún:

> **luciano:** [...] la idea de la ética *hacker* y del conocimiento libre, de que yo no soy dueño de lo que tengo dentro de mi cabeza, sino que ha sido a base de construcción, de estar parado en hombros de gigantes. Y por eso es algo que le pertenece a la humanidad entera.

Y

> **karora:** Because part of what you believe if you work on free software, part of what you most likely believe in, is the social ownership of the work you are doing. You are essentially doing something to be given away, to make the world a better place.

Y es que el concepto de procomún es esencial para evitar las paradojas de la consideración del *software* libre como don, para entender que la obligación de devolver sólo se entiende a partir de la relación con el colectivo, no entre individuos. En un mensaje de un hilo que ya mencionamos en otro contexto (en la sección 3.3, *Debian como público recursivo*) encontramos la siguiente afirmación:[33]

> I think you're working from a different definition of gift than I was, and I probably muddled the waters by talking so much about personal gifts to specific people. Free software isn't that sort of gift; it's a gift to the public, which is a different sort of thing, since it doesn't create those sorts of personal relationships. It's a gift to strangers, a way of saying «I made this thing, and if you find it useful, you can use it too.»

5.2.4.1 Lo imaginario y lo simbólico: lo sagrado y la cuarta obligación del don.

Godelier se preguntaba por las razones de esa inalienabilidad, y las encuentra en la propia supervivencia del colectivo:

> [...] los grupos humanos se hayan esforzado por preservar sus condiciones de existencia (materiales o no, pero siempre reales a sus ojos), por sustraerlas a la dispersión, a la partición y al desmembramiento, otorgándoles el carácter de un bien que debe conservarse para ser

[33] En Debian-devel (2016c). En ese hilo hay mensajes muy interesantes sobre esta cuestión, y sobre las obligaciones que crea el don. Véase por ejemplo Debian-devel (2016d) para una interesante analogía que compara el *software* libre con una fruta que se toma de un árbol.

> transmitido tal cual, indiviso, y asegurar así la vida, la supervivencia de las generaciones futuras (Godelier, 1998, pág. 71).

Como ocurre en el *software* libre, para que se dé esta continuidad temporal tiene que haber en la cosa donada algo más que el don de uno mismo, la presencia del donante:

> Es preciso que contenga *además* alguna cosa que a *todos* los miembros de la sociedad les parece indispensable para su existencia, y que debe *circular entre ellos* para que todos y cada uno puedan seguir existiendo (Godelier, 1998, pág. 108).
>
> Junto con el propietario, lo que está presente en el objeto es todo el imaginario de una sociedad, de su sociedad. En él residen todos los dobles imaginarios de los seres humanos, a los que estos últimos atribuyen (no puede decirse que se los presten, puesto que esos dobles nada pueden devolver) los poderes para reproducir la vida, para acarrear salud y prosperidad o bien sus contrarios, la muerte, el hambre y la extinción del grupo (Godelier, 1998, págs. 40-1).

Por esto en el *software* libre se expresa también todo el imaginario social relativo a una condición fundamental para el desarrollo y la supervivencia de las sociedades contemporáneas: la información y el conocimiento, que como vimos en la formulación clásica de la ética *hacker* «quieren ser libres» (véase la sección 4.2.2, *Mediadores e intermediarios. Las figuras del* bug *y del* hacker), es decir circular libremente. Como estamos viendo en esta sección y como aparece en Weiner (1992), hay un vínculo profundo entre las posesiones inalienables y la identidad del grupo, también sus condiciones de existencia. De ahí también el papel fundamental de los objetos en la constitución del colectivo. En Godelier (1998, págs. 76-85) encontramos una extensa explicación de cómo el concepto de *hau* usado por Mauss es también el *hau* del bosque. Ahí se cita, completándola (y usando una nueva traducción) la explicación sobre el *hau* de Tamati Ranaipiri citada por Mauss:

> Voy a explicarte una cosa sobre el *hau* del bosque. El *mauri* ha sido colocado o implantado en el bosque por los *tohunga* [los sacerdotes]. Es el *mauri* que hace aumentar el número de pájaros en el bosque, a fin de que el hombre pueda matarlos y capturarlos. Esos pájaros son propiedad de los *mauri*, de los *tohunga* y del bosque. *Les pertenecen*. Dicho de otro modo, son un equivalente de esa cosa considerable, el *mauri*, y es por eso por lo que se dice que hay que hacer ofrendas al *hau* del bosque. Los *tohunga* se comen la ofrenda porque el *mauri* [la piedra sagrada] es suyo. Son ellos los que lo han emplazado en el bosque, los que le han hecho ser. Por esta razón, algunos de los pájaros cocidos sobre el fuego sagrado son apartados para alimentar a los sacerdotes y sólo a ellos, a fin de que el *hau* de los productos del bosque y el *mauri* retornen nuevamente al bosque, es decir, al *mauri*. Ya basta de esto (Godelier, 1998, págs. 78-79).

Y más adelante:

> Dos ideas se hallan asociadas en el ejemplo del bosque, los sacerdotes y los cazadores. La primera es que el bosque es fuente de vida y de multiplicación de la vida. Es el bosque quien, finalmente, concede las piezas, en forma de don, a los cazadores. La segunda es que las piezas capturadas por los cazadores no han dejado de pertenecer al bosque y a los sacerdotes que poseen tanto el objeto sagrado como la fórmula que lo acompaña y permite incitar al bosque a mostrarse generoso con los humanos (Godelier, 1998, pág. 80).

Resulta tentador hacer la analogía entre el papel del bosque como tal en esta descripción y el del procomún del *software* libre tal y como lo venimos utilizando. Sin desarrollar las posibles correspondencias entre los sacerdotes y los Desarrolladores, o entre las piedras sagradas y determinados componentes del *software*, sí que se puede señalar la importancia de esa fuente de recursos que necesita ser cuidada, y que es para los cazadores y los usuarios la fuente de los dones. Recordemos los fragmentos de entrevista citados en la sección 5.1.2, *¿Por qué se dona el*

software *libre?*, sobre la necesidad de devolver, que la entendían como el cuidado de esos recursos comunes. El bosque y el procomún del *software* libre son elementos necesarios, así como sujetos, en la creación, distribución y circulación de los dones, son parte necesaria de las cadenas de prácticas asociadas. Son «fuente de vida y de multiplicación de la vida». Tienen un importante papel productivo, no sólo en sentido material.

A este respecto, ya hemos hablado en este capítulo y los anteriores de la importancia de lo imaginario, en un sentido general, para la constitución de los colectivos. Podemos intentar ahora, aunque sea brevemente, situarlo en la oposición clásica a lo simbólico como dos órdenes que constituyen lo social. Godelier critica la tesis (proveniente de Lacan[34]) de Lévi-Strauss sobre la superioridad de lo simbólico frente a lo imaginario respecto de la explicación de los hechos sociales. Lo imaginario, «las diferentes maneras que los hombres tienen de imaginar sus relaciones, entre sí y con eso que llamamos naturaleza» (Godelier, 1998, págs. 46-47), remite, en Mauss, a lo sagrado y a su papel constitutivo de la sociedad. En la prioridad de lo simbólico en Lévi-Strauss lo que encontramos es la prioridad explicativa y constitutiva del intercambio y las relaciones, y de las leyes que lo gobiernan.

Tomar entonces esta analogía en serio sugeriría entonces al menos dos cosas. La primera nos llevaría a entender el contenido de la dimensión de lo imaginario en el colectivo del *software* libre como las representaciones sobre el compartir, la reciprocidad o la comunidad que estos hacen posible, mientras que la dimensión de lo simbólico haría referencia a los dispositivos de licenciamiento y de apertura del código.

De ahí la presencia del componente utópico en muchas representaciones de la importancia y el significado del *software* libre en el mundo contemporáneo. Aunque Godelier (como Mauss) está pensando en el ámbito individual y sobre todo en el don caritativo, podemos reconocer lo que dice hacia el final de su libro:

> [en las sociedades modernas occidentales] Al idealizarse, el don «sin cálculo» funciona en el imaginario como el último refugio de una so-

[34] No entraremos en la tripartición del psicoanálisis lacaniano: lo imaginario, lo simbólico y lo real.

> lidaridad, de una generosidad en la distribución, que habría caracterizado a otras épocas de la evolución de la humanidad. El don se hace portador de utopía (una utopía que puede proyectarse tanto hacia el pasado como hacia el futuro) (Godelier, 1998, pág. 296).

Podríamos distinguir así toda una serie de dualismos encadenados: imaginario y simbólico (en esta misma sección), orden social y orden técnico-legal (sección 3.3, *Debian como público recursivo*), expresivo e instrumental (sección 4.3, *Instrumentalidad y expresividad del código*), lo social y lo técnico (sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*), el *software* libre y el *software* de código abierto (sección 5.2.2, *Sobre el sentido de «libre» en el* software *libre y la cultura libre*), entre otros.

Pero como se viene insistiendo a lo largo de todo este trabajo, de lo que se trata es precisamente de evitar caer en el apriorismo de esos dualismos, estudiando cómo son construidos en las prácticas de los sujetos. Godelier se plantea el problema en términos de prioridad explicativa bien de lo imaginario, bien de lo simbólico. Pero de lo que se trata aquí es de la constitución del colectivo, situando sobre un mismo plano todos estos elementos para poder seguir sus conexiones y vínculos. No hay más contenido social que la circulación y vinculación de elementos heterogéneos (Latour, 2005). Los elementos caracterizados como parte de lo imaginario son por supuesto importantes en esta descripción, pero en tanto son producidos y movilizados por los agentes en sus prácticas, no como un contenido o substancia más propiamente «social». La distinción entre lo imaginario y lo simbólico depende de la semiótica saussureana y su noción dualista del signo, mientras que como venimos viendo se explica mucho mejor la producción de estos dualismos a partir de los conceptos de la semiótica de autores como Peirce y Kockelman, y su énfasis en la agencia y los procesos.[35]

La segunda sería la sugerencia de que el procomún ocupa de alguna manera el lugar de lo sagrado. Es respecto a este ámbito que Mauss formula la cuarta obligación del don:

[35] Para una contraposición entre ambos tipos de semiótica, véase Díaz de Rada (2013).

> donar a superiores no implica necesariamente que estos últimos sean seres humanos. En todas las sociedades –estén o no divididas en rangos, castas o clases–, vemos cómo los humanos realizan dones a seres que consideran superiores, a los poderes divinos, a los espíritus de la naturaleza o de los muertos [...] Se trata de la famosa «cuarta obligación» constitutiva del don (Godelier, 1998, págs. 26-27).

Aquí parece que nos alejamos demasiado del material empírico de esta investigación, en tanto que no hemos encontrado referencias a conceptos o nociones de tipo religioso. Nuestro campo de investigación aparece como completamente ajeno a este ámbito, y es aquí donde se encuentra el punto de mayor distancia entre el complejo de acciones a los que tradicionalmente se han aplicado los conceptos antropológicos desarrollados en torno a los fenómenos del don, y las cadenas de prácticas que constituyen los colectivos donde se produce y distribuye *software* libre. Pero más que el contenido concreto de la noción de lo sagrado, que está muy lejos de estar claro y sobre todo de ser unívoco (Casajus, 2005), nos interesa aquí su lugar estructural en relación a la circulación de los dones.

Y en efecto, encontramos en el procomún la noción de un ámbito que ha de ser protegido y que de alguna manera es tanto el punto de origen como el de destino de la serie de dones, esencial para la supervivencia del colectivo, y respecto al que se imaginan y formulan la identidad y los principios constitutivos del mismo, así como el ámbito respecto del que podemos predicar la inalienabilidad de los dones. Ya hemos visto cómo se constituye en un dominio central, en tanto que los dones que estamos estudiando no se dirigen a sujetos concretos, sino que donarlos supone precisamente situarlos en ese espacio del procomún. En ese sentido, se puede decir algo análogo a lo que afirma Mauss sobre los «espíritus de los muertos y los dioses»:

> por otra parte, Mauss afirma que los espíritus de los muertos y los dioses «son los auténticos propietarios de las cosas y bienes del mundo. Es con ellos que resultaba más necesario intercambiar, y más peligroso no hacerlo» (Godelier, 1998, pág. 50).

El lugar central del procomún tiene también como consecuencia la desproporción entre lo que uno puede donar y lo que recibe a cambio, como ocurre también en el intercambio de dones con el ámbito de lo sagrado:

> [estos sacrificios], como escribe Mauss, «llevan hasta el extremo» la economía y el espíritu del don, pues «esos dioses que donan y devuelven están ahí para donar grandes cosas a cambio de otras pequeñas» (Godelier, 1998, pág. 51).

Para entender hasta dónde puede llegar esta analogía se hace entonces necesario introducir la cuestión de la gracia. Y es que los bienes que analizamos poseen una propiedad particular en tanto que al ser digitales son infinitamente copiables. Aparece así una capacidad de donar infinita, mediante la que se puede dar sin perder eso que se da. En este ámbito la donación de bienes no se puede asimilar ni hacer depender de la reciprocidad, porque ésta supone precisamente que no hay capacidad infinita de donación. Encontramos así una vinculación con el concepto teológico de la gracia. Existe gracia precisamente cuando no hay reciprocidad posible, cuando lo donado aparece como exceso gratuito respecto a las posibilidades de retorno (Pitt-Rivers, 1992, pág. 216). Así aparece precisamente el procomún, como una fuente de dones que siempre excederán aquello que se puede devolver, como un sujeto precisamente porque excede toda reciprocidad entre individuos. Pero al mismo tiempo, como algo a lo que en definitiva uno puede contribuir, aumentando su riqueza y capacidad productiva.

Los sentidos del concepto de gracia son múltiples: se usa para agradecer, para indicar la ausencia de precio, o para referirse a lo que sólo Dios puede donar, por citar algunos ejemplos. Lo que parece unificarlos es esa dimensión de exceso:

> The only general rule that can be cited is that grace is always something extra, over and above «what counts,» what is obligatory or predictable; it belongs on the register of the extraordinary (hence its association with the sacred) (Pitt-Rivers, 1992, pág. 217).

Y según Pitt-Rivers, podemos ligar su sentido a algunas de las oposiciones que venimos viendo en esta sección.

> There are, then, two parallel modes of conduct, [...] the sacred and the profane. They are governed, respectively, by the principle of grace and by the principle of law, that is to say, predictable regularity, as well as justice and the law which impose order in human affairs – from which pardon (grace) authorizes a departure. Under the heading of «grace» it is possible to group all the phenomena that evade the conscious reasoned control of conduct (Pitt-Rivers, 1992, pág. 221).

Esta formulación establece un dualismo difícilmente aceptable después de todas las consideraciones que venimos haciendo. Ambas dimensiones no son excluyentes, y podemos hablar de dos dimensiones de los fenómenos citados, o dos dimensiones de las prácticas que son inseparables más que de dos tipos de conducta. En cualquier caso, el concepto teológico de la gracia como el don gratuito e inmerecido de Dios es el relevante aquí, puesto que es el que nos permite apreciar por analogía la aparición del procomún como un sujeto y agente por derecho propio en la cadena de dones y prácticas. Un sujeto que por supuesto sólo actúa en asociación con el resto de agentes, como por otra parte opera también la gracia:

> It develops within us as an *habitus*, an acquired disposition to cooperate with the will of God, and this involves human will also, upon which the will of God operates, in St. Augustine's opinion. [...]
>
> [...] the essence of grace is the will of God, which necessarily restricts the individual's will in some degree, but the attainment of grace can only be achieved with the cooperation of human will since God requires his beneficence to be returned (Pitt-Rivers, 1992, pág. 222).

Como el mismo Pitt-Rivers señala, no nos hemos alejado demasiado de la noción de *hau*:

> many of the senses of this multi-faceted concept (though by no means all) are analogous with either grace, *baraka* or *indarra*, all of which are highly varied, as we have seen. One finds in them the essence of the gift, its non-contractual nature and something like the soul of the donor contained in it. *Hau*, like grace and like *baraka*, is that which is in excess, left over or supplementary, transcending exact reciprocity. [...]
>
> I am not pretending that *hau* can be translated as grace, *baraka*, or *indarra*, but only that it is a concept of the same order, with all the differences that are implied by the contrast between a monotheistic and a polytheistic mode of thought. [...] so I can see that *hau* can be, at least in many contexts, assimilated to grace (Pitt-Rivers, 1992, pág. 237).

La gracia y el *hau* comparten entonces ese carácter de exceso creativo, que por eso mismo es capaz de hacer surgir nuevas formas de acción y de agencia. Se convierten en componentes fundamentales de la acción. Si bien es cierto que esta serie de nociones de tipo religioso aparecen alejadas en cuanto a su contenido de las de procomún, libertad o apertura, lo interesante, como decíamos más arriba, es su lugar estructural. Y es que ese exceso de capacidad creativa no surge sino de la asociación de los principios y objetivos expresados en declaraciones como el Contrato Social de Debian o las DFSG, dispositivos legales como las licencias, y técnicos como el archivo. Como la gracia, el *hau* o lo sagrado respecto del sacrificio, los dones y sus reglas, el procomún y el carácter libre del *software* no se dejan reducir a esos dispositivos, pero tampoco se pueden pensar o describir al margen de ellos.

Veamos ahora un fragmento de una entrevista que se entiende mejor después de lo que hemos expuesto en esta sección:

> **micah:** I don't know, there is also a strong affinity with people who are Debian users, once you become a Debian ...., converted to Debian, you are almost like part of a cult or something [risas]. You realize how great it is and you love it. And you see Debian users, they start to love Debian, you know, and so do the Developers, because they get so much

> use out of it. [...] They get a lot of benefit from whatever this community is doing, and they continue doing their work with it, because they are getting so much benefit. And because they get so much benefit, they are expressing their gratitude and their appreciation, and that is some kind of fuel into this community to keep doing what they are doing.

5.2.4.2 *Keeping-for-Giving-and-Giving-for-Keeping*: la conservación de la libertad del *software*.

En definitiva, el concepto de procomún permite entonces pensar la posibilidad de un espacio de producción liberado de las constricciones impuestas por el régimen de propiedad intelectual, al tiempo que permite observar el surgimiento y consolidación de prácticas relacionadas con la creación, distribución y circulación de diferentes formas de conocimiento. Este proceso supone entonces una serie de innovaciones que residen sobre todo en el papel de las licencias, que regulan esas condiciones de distribución y de uso. Si el procomún digital abre un espacio liberado, lo hace precisamente utilizando las constricciones establecidas por el régimen establecido de propiedad intelectual.

Aparece así un espacio nuevo, con capacidad de agencia diferenciada, en tanto que es el sujeto al que se dona y que mantiene determinados derechos sobre esos dones. Usado en este sentido, es un término inseparable de los de comunidad y colectivo (véase el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*).

Este ámbito es frágil y surge la necesidad de conseguir su perpetuación, de protegerlo de la apropiación privada. Precisamente esta necesidad de protección es lo que permite a O'Mahony (2003) establecer que el *software* libre es un bien común (procomún) y no solamente un bien público. Ésta es precisamente la característica que los distingue, y que frente a las apariencias afecta al *software* libre:

> we have established that open source and free software is privately produced and non-exclusive. This meets the definition of both public goods and common pool resources. The next dimension is more ambiguous: is open source software subtractable or joint in supply? If one

> person downloads software for his or her personal use, the amount available for the next person is unchanged. However, it requires more protections than those offered by the public domain. To remain open and publicly available, it must be protected from proprietary appropriation. Thus, open source and free software appear to be joint in supply, but are in fact vulnerable to usage that would threaten its availability to all. Use of the software will not diminish in the present, but the future stream of benefits is at risk (O'Mahony, 2003, pág. 1182).

El procomún del *software* libre necesita entonces de mecanismos[36] para protegerlo, para mantenerlo, en los términos de Weiner, inalienable.[37] Lo que en el caso del *software* libre y la cultura libre quiere decir mantenerlos inalienables *del procomún*, a través de las licencias. Si una de las tesis fundamentales de Weiner es que existen (incluso en «economías del don») siempre cosas que no se donan y no circulan (*Keeping-while-Giving*), que permiten la afirmación de la identidad y su continuidad en el tiempo, así como el establecimiento de jerarquías,[38] Godelier pretende «ir más allá, hasta señalar que una forma más adecuada sería *Keeping-for-Giving*, guardar para (poder) donar». En nuestro caso, esto implica licenciar con una licencia libre (guardando para la comunidad) para que el *software* pueda ser compartido, pueda circular. Lo que implica al mismo tiempo «donar para (poder) guardar», es decir, liberarlo (liberarlo aquí quiere decir publicarlo con una licencia libre) para que siga estando siempre disponible para la comunidad:

> Así pues, la fórmula de lo social no es *Keeping-while-Giving*, sino *Keeping-for-Giving-and-Giving-for-Keeping*. Guardar para (poder) donar, donar para (poder) guardar (Godelier, 1998, pág. 58).

[36] O'Mahony (2003) detalla varias estrategias al respecto, alguna de las cuales son también consideradas en esta investigación.

[37] Sobre la necesidad de protegerlo, desde una perspectiva crítica, véase también Ossewaarde y Reijers (2017), donde se examina la aparición de una cierta «ilusión del procomún digital» y la necesidad de superar la misma para desarrollar su potencial emancipatorio.

[38] Trataremos es asunto de las jerarquías y la desigualdad más adelante en este mismo capítulo. Según Godelier (1998, págs. 54-55), la segunda idea esencial de Weiner tiene que ver con el papel de las mujeres. Aquí no tratamos esta cuestión puesto que, aunque en el *software* libre y en Debian existen desigualdades referidas al género, éstas no dependen (como en los casos analizados por las teorías del don) de su relación con el parentesco.

En el caso del *software* libre, esto es un solo movimiento: licenciar con una licencia libre. Lo que permite circular libremente al código es lo mismo que lo mantiene protegido en el procomún impidiendo su apropiación privada. Si Weiner establecía la existencia de dos clases de posesiones, alienables e inalienables, lo que encontramos en este caso son dos aspectos o dimensiones de los mismos objetos más que dos clases diferentes de objetos. Así lo expresa O'Mahony (2003):

> Observers of the open source phenomena questioning why contributors to community managed projects would give their work away for free have neglected to examine what is given away (code) and what is retained (rights) (O'Mahony, 2003, pág. 1180).

Lo que hace una pieza de código inalienable en ese sentido es entonces su licenciamiento. Pero no olvidemos que la posibilidad de añadirle una licencia determinada depende de su creador. Es el hecho del vínculo entre desarrollador (o desarrolladores) y código lo que permite que se convierta en patrimonio del procomún. Weiner afirma:

> What makes a possession inalienable is its exclusive and cumulative identity with a particular set of owners through time. Its history is authenticated by fictive or true genealogies, origin myths, sacred ancestors, and gods. In this way, inalienable possessions are transcendent treasures to be guarded against all the exigencies that might force their loss (Weiner, 1992, pág. 33).

Es fundamental entonces establecer y mantener la genealogía de lo que se quiere preservar. En la práctica, como vimos en las secciones 4.5, *El empaquetado como práctica sociotécnica*, 4.8, *Repositorios de* software *libre* y 5.2.3, *Las licencias en Debian*, es necesario mantener los registros de autoría.[39] En este sentido, y a diferencia de muchas de las situaciones analizadas por las teorías clásicas del

[39] Que pueden ser extensos y complicados, convirtiéndose en verdaderas genealogías. Esto puede ayudar de hecho a mantener algunas obras en el procomún: el código es más difícil de privatizar (cambiar su licencia) si existen muchos autores con los que ponerse de acuerdo.

don, en el campo del *software* libre y en Debian sí que se llevan las cuentas (el registro) de las donaciones de cada uno (al menos las contribuciones al código). No está tampoco mal visto que sea uno mismo quien lo haga y lo publicite. Pero al ser donaciones al procomún no generan desigualdad y jerarquía en el mismo sentido que esas otras situaciones, como veremos más adelante.

Una cuestión que surge en este punto es el hecho de que la asignación, el cuidado y el mantenimiento del *copyright* sobre los elementos de *software* sigue siendo individual, a pesar de la dimensión colectiva o al menos supraindividual que hemos visto en el procomún. Esto significa que la responsabilidad de defender no ya los propios derechos del autor, sino los derechos del usuario, recae sobre el autor individual. Lo que en la práctica puede suponer que estos derechos no sean defendidos, debido a la carga que supone, en los tribunales en última instancia, cuando no son respetados.

Para facilitar esta defensa, en la DebConf de 2015 se presentó el *Debian Copyright Aggregation Project*, un programa de la *Software Freedom Conservancy* que permite asignar el *copyright* a esta organización, o alternativamente el derecho a defender las condiciones impuestas por la licencia. Esta organización sin ánimo de lucro

> helps promote, improve, develop, and defend Free, Libre, and Open Source Software (FLOSS) projects. [...]
>
> Conservancy provides many important services for its member projects. Member projects can take directed donations, [...]
>
> [...] Conservancy can also hold other assets and titles on behalf of the projects, such as copyrights, trademarks, domain names, online hosting accounts, and title and ownership of physical hardware. Also at discretion of the project's leaders, Conservancy can assist in defending the rights represented in these assets. For example, Conservancy is available to assist member projects in enforcing the terms of the projects' FLOSS license. [...]
>
> Conservancy and its directors, officers, and staff believe strongly in the principles of software freedom, and believe that all users should ha-

> ve the right to study, improve and share their software. Conservancy helps protect, enable, coordinate, facilitate and defend the public's right to copy, share, modify and redistribute FLOSS both non-commercially and commercially (Sfconservancy.org, s.f.[a]).

Esta presentación la realizó Bradley Kuhn, de quien hemos hablado anteriormente, presidente de esa organización.[40] El *Debian Copyright Aggregation Project* es un programa voluntario, y la *Software Freedom Conservancy* mantiene programas similares con otros importantes proyectos de *software* libre.

No es un programa para la protección del derecho de los autores, sino de los derechos de los usuarios, aunque para ello necesiten la colaboración, la cesión, de los autores. El objeto de su defensa, como indica el nombre de la asociación, es la libertad del *software* y la libertad de la comunidad («As stalwarts of the community's freedom, we act as a proxy for users when companies impede the rights to copy, share, modify, and/or redistribute copylefted software»), como se manifiesta en «The Principles of Community-Oriented GPL Enforcement» (Fsf.org, s.f.[c]; Sfconservancy.org, s.f.[b]). En inglés, el término *conservancy* remite a organizaciones que trabajan en la conservación de espacios y recursos naturales, de interés público. El mismo título de este *post* de la *Software Freedom Conservancy* muestra que la defensa del cumplimiento de la GPL (u otras licencias *copyleft*) se hace por los intereses de la comunidad. En la misma página se establece también: «The GPL, enforced when necessary according to these principles, provides a foundation for respectful, egalitarian, software-sharing communities». A fin de cuentas, los derechos que protege un autor al usar una licencia *copyleft* son los derechos a que el *software* se siga distribuyendo a los demás.

Se establece también ahí claramente que el objetivo de éste y otros programas similares es el respeto general a la GPL y la generalización de la libertad del *software*, no obtener compensación por sus violaciones. Lo fundamental es lo que es beneficioso para la comunidad, que el *software copyleft* siga siéndolo para la

[40] Se puede ver la presentación en Meetings-archive.debian.net (s.f.[d]). El anuncio en la página de Debian se puede leer en Debian.org/News (2015), y en la de *Software Freedom Conservancy*, en Sfconservancy.org (2015). Un resumen de la presentación se puede leer en Willis (2015).

comunidad, no asegurarse de que los autores reciban algo a cambio. Esto es congruente con lo que afirma Weiner:

> As a generic concept, reciprocal exchanges are only the pawns on the chessboard of culture preserving inalienable possessions and fending off attempts by others to claim them (Weiner, 1992, pág. xi).

Encontramos así una encarnación de la fórmula de Godelier: *Keeping-for-Giving-and-Giving-for-Keeping*. Proteger el *copyright* para que el *software* siga circulando libremente, y donar para mantener el *software* en el procomún, en el colectivo. También posiblemente donar el propio *copyright* (o la capacidad para aplicarlo) para que sea defendido y mantenido eficazmente.

El requerimiento de algunas empresas de que sus empleados le asignen el *copyright* del *software* que desarrollan incluyendo el *software* libre ha sido problemático en ocasiones. La diferencia con lo que estamos viendo es que aquí se solicita que se asigne a una organización benéfica sin ánimo de lucro. En inglés, a una *charity*. Las connotaciones del concepto son más amplias en inglés que en castellano, y como vimos más arriba Debian depende de ellas (sus *Trusted Organizations*) para manejar sus bienes y sus activos. Lo siguiente es una cita de una diapositiva de la presentación de Kuhn:

> Thus, the charitable tendencies of GNU, FSF, Debian and (more recently) SPI and Conservancy were and remain perfect fit for creating, fostering, and defending democratic, developer-run community projects which are (mostly) under copyleft licenses.

En el contexto, *charitable* está también opuesto a «profesional», pero nos lleva de nuevo al concepto de gracia, con el que comparte la raíz etimológica (Pitt-Rivers, 1992, pág. 221), y con él a la importancia estructural de la comunidad y el procomún. También a la importancia dada por Godelier y por Mauss a la recuperación de la caridad en las sociedades contemporáneas.

A diferencia de otros bienes, para que funcione y siga siendo útil y valioso, el código es necesario no sólo mantenerlo, sino también desarrollarlo continuamente. Incluso si no se quieren añadir cambios o funcionalidades nuevas, simplemente para corregir los fallos (*bugs*) y adaptarlo a entornos nuevos o en evolución. Y cada cambio de este tipo requiere un esfuerzo para mantenerlo en el procomún.

En cualquier caso, en el caso del código que se encuentra en el procomún, mantener su disponibilidad (el doble movimiento que supone guardarlo (*keeping*) y hacerlo circular, donarlo (*giving*)) supone enriquecerlo, desarrollarlo, aumentarlo. Como dice Godelier en la cita que abre este capítulo:

> Más allá de la esfera de los intercambios, existen otros dominios, otra esfera constituida de todo aquello que los hombres imaginan que deben sustraer al intercambio, a la reciprocidad, a la rivalidad, de todo aquello que creen que deben conservar, preservar, incluso enriquecer (Godelier, 1998, pág. 58).

Así pues, como los objetos que circulan en el *kula*, el procomún digital no sólo hay que protegerlo, también hay que producirlo. Así explica Godelier[41] la noción de *kitoum*, referida a los objetos que circulan en el *kula* y que son inalienables:

> Para comprenderla, es preciso volver a partir del hecho de que todos los objetos que circulan en el *kula* (brazaletes y collares) son objetos fabricados [...] se convierten entonces en un objeto de valor, propiedad personal de quien lo ha fabricado, en su *kitoum* (Godelier, 1998, pág. 132).

Y así lo hace Strathern:

> [...] the class known as *kitoum*, which are those *kula* valuables a man deploys by virtue of their standing for some part of his activity [...] They

[41] Destacando su descubrimiento simultáneo por parte de Weiner en Kiriwina y de Damon en Muyuw.

> speak of *kitoum* as objects they have manufactured (Strathern, 1990, pág. 195).
>
> Conversely, Muyuw conceptualize *kitoum* as "productive" items (e.g., Damon 1983a:285). They are apprehended as things that can be used to make something else. *Mwal/veigun* on the other hand are regarded as 'finished articles' (Strathern, 1990, pág. 370).

Godelier (1998, pág. 139) distingue dos tipos de propiedades que comparten el ser inalienables: el *kitoum* individual, producido por un individuo (y que puede ser donado); y la tierra por ser un bien común. Ambos tipos quedan condensados en lo que llamamos procomún digital del *software* libre: éste está formado por productos individuales donados por sus productores, que por ese acto de donación y sus características especiales (su licencia) pasan a ser un bien común. Curiosamente, el *kitoum* que circula en el *kula* se puede donar «con la condición de que el objeto poseído no salga del marco del *kula*, ni sirva para otro uso que el de actuar como don o como contradón» (Godelier, 1998, pág. 135). El carácter *copyleft* de la GPL pretende también justamente eso, que el *software* modificado pueda seguir donándose pero no ser apropiado privadamente.

En definitiva, una característica esencial de lo que no es alienable (y que explica que no sea alienable) es el hecho de ser producido y de ser necesario para producir más, su productividad y creatividad. Lo que nos conduce desde la segunda a la tercera obligación del don.

5.3 La obligación de recibir.

Como ya hemos visto, para Mauss el fenómeno del don está constituido por tres obligaciones complementarias: la de *dar*, la de *devolver*, y la de *recibir* (Mauss, 2009, pág. 91). Ya hemos hablado extensamente de las dos primeras, que son también las más tratadas en general por la mayoría de autores. En el caso del *software* libre, la primera aparece normalmente vinculada a las motivaciones que llevan a los desarrolladores a ofrecer el producto de su trabajo, mientras que la segunda queda asegurada por el mecanismo de las licencias. El uso de las teorías del don para la comprensión del campo que estamos investigando se suele centrar

en estas dos cuestiones. Ghosh, por ejemplo, sitúa las similitudes entre sociedades «tribales» y el campo de la economía digital respecto a la dinámica de los dones en esos dos aspectos:

> The giving of gifts in tribal societies is, in reality, an action performed within the context of reciprocity and expectations of returns –status, rights, or more gifts. There are indeed similarities between collaborative production and nonmonetary exchange in tribal societies and collaborative ownership in the digital economy, notably free software: both are based on the self-interested participation of individuals and communities linked by a complex web of rights and obligations (Ghosh, 2005b, pág. 7).

Se entiende el don así como inversión, normalmente en busca de capital simbólico o social (Ortega y Rodríguez, 2011; Zeitlyn, 2003). Además de las motivaciones (la obligación de dar), se puede establecer la analogía a partir del mecanismo que mantiene los bienes en circulación (la obligación de devolver). Así, más importante que la motivación para dar, es el hecho de que hacerlo supone sobre todo entrar (y hacer entrar el producto donado) en una red de relaciones, en la que las licencias libres mantienen a los bienes en circulación. Pero como señala Boyle (2005) a propósito de la *General Public License*:

> The point, however, is that the open quality of the creative enterprise spreads; it is not simply a donation of a program or a work to the public domain, but a continual accretion in which all gain the benefits of the program on pain of agreeing to give their own additions and innovations back to the communal project (Boyle, 2005, pág. 244).

Cuando hablamos del procomún y del *software* libre, no se trata exclusivamente de la existencia de un ámbito y unos repositorios que ofrecen sólo productos ya terminados. Como Boyle señala, lo verdaderamente importante está más en su crecimiento continuo que «simplemente en la donación de un programa o una obra». Como hemos visto en la sección anterior, esta potencia productora y

creativa del procomún es esencial. Este crecimiento depende por supuesto de la obligación de devolver que imponen las licencias, pero también de la existencia de dispositivos y prácticas que permitan la contribución a proyectos en desarrollo. Veamos a continuación cómo se implementan estos dispositivos y prácticas.

5.3.1 El crecimiento del procomún y la obligación de recibir.

La tercera obligación relativa al don postulada por Mauss, la de recibir, ha sido entonces la que menos atención ha recibido, también en su aplicación a las nuevas formas de producción colaborativa o a la cultura libre. Vamos ahora a considerar si existe, y cómo se encarna, esta tercera obligación en el campo del *software* libre y el procomún digital.

Para ello, debemos centrarnos en los procesos que constituyen el procomún del que hemos hablado en secciones anteriores. Los derechos que normalmente se han enfatizado en relación a este ámbito son los derechos a tomar algo de él, más que a contribuir al mismo, y a que esta situación se mantenga en el tiempo. Es decir, se suele pensar en el procomún desde la perspectiva de *cómo protegerlo*.[42] Aquí el modelo son los bienes comunes «naturales»,[43] y la cuestión central es por supuesto el uso de licencias libres que permiten precisamente eso, como hemos visto en las secciones anteriores. Esta perspectiva permite a los participantes en estas prácticas afrontar la aparición de nuevas tecnologías y disposiciones legales que posibilitan la apropiación de ideas, dispositivos y procesos que de hecho eran bienes comunes antes de su aparición (Boyle, 2005; Lessig, 2005).

Pero entender las prácticas que estamos estudiando, y cómo constituyen un espacio procomunal, requiere además una perspectiva complementaria: centrarse en los procesos que posibilitan que este espacio crezca. Procesos que convierten las mediaciones legales y tecnológicas en capacidad –y obligación– de recibir. Esto pasa a su vez por ver cómo se aseguran de alguna manera los derechos de la gente a contribuir a ese espacio y no sólo a tomar de él. Y no sólo añadiendo más recursos ya terminados, de manera independiente, sino colaborando activamente en el

[42] Uno de los ejemplos más claros o más explícitos es el de Boyle (2005, 2008), que trata la cuestión a partir de la amenaza de una segunda *enclosure* o cercamiento.
[43] O lo que Lafuente (2007) llama el entorno «medioambiente» del procomún.

mantenimiento y mejora de los que ya existen. A diferencia de los bienes comunes tradicionales (o mejor dicho, a diferencia de cierta visión economicista sobre ellos), el procomún digital no es un recurso natural, ya listo y disponible para explotar. Además de cuidarlo y protegerlo, hay que producirlo continuamente. Este aspecto distingue también el *software* libre de otros campos del procomún digital (más ligados a objetos artísticos, y más relacionados con las licencias *Creative Commons*): se puede contribuir efectivamente a desarrollos en curso, y no limitarse a utilizar lo que ya está hecho.

Ésta es también una perspectiva necesaria a partir de la consideración que venimos realizando del procomún como un sujeto, puesto que la obligación y la capacidad de recibir se predican de un sujeto. En el caso que estamos estudiando, la obligación de recibir sería una obligación cuyo sujeto es colectivo, o al menos no individual ni discreto: es el procomún el que recibe. Posiblemente, aquí esté de hecho la esencia del *software* libre (y que lo opondría a cosas como el *freeware*, la simple gratuidad del *software*). Si hablamos de cada comunidad, de cada proyecto de *software* libre en concreto (como por ejemplo Debian), está obligación es en todo caso muy limitada, en tanto que no se ven forzados a admitir mejoras o modificaciones en la propia versión que se ofrece. Pero sí es cierto que tienen que permitir las mejoras y modificaciones que cualquiera pueda producir al menos como *fork*, como producto derivado y paralelo, que desde luego entrará a formar parte del procomún. La obligación de recibir no es mera condición lógica de las obligaciones complementarias de donar y devolver, sino que se convierte aquí en una condición de la aparición del procomún como sujeto productivo y creativo, como constituyente fundamental del colectivo formado en torno al *software* libre. En todo caso, en lo que sigue hemos de tener en cuenta que la obligación de recibir se establece no entre individuos particulares, sino entre ese entorno que llamamos procomún y las aportaciones de un individuo.

En la teoría antropológica clásica del don, la obligación de recibir es un complemento a la obligación de dar que está relacionada con la producción de jerarquía y desigualdades, a través de su capacidad de crear deuda. Dejamos esta cuestión para más adelante. Pero en Mauss se basa también en el hecho de la copresencialidad, que introduce una coacción pragmática de la que es difícil escapar. Al ser una relación entre personas, más allá de la que se da entre objetos como en el

caso de la mercancía, el don se ve reforzado por la presencia mutua de los sujetos. Los dones se ofrecen en persona, y al mismo tiempo es más difícil rechazar un don frente a la presencia física del donante. La distancia y la separación no harían sino dificultar el don.

Mackenzie plantea una cuestión parecida en un contexto tan diferente como es el de los mercados financieros. El contexto es diferente porque en éste se trata de maximizar el beneficio económico en situación de competencia con los demás actores. Pero en ambos encontramos un problema de acción colectiva, en el que las condiciones para mantener la confianza difieren entre la copresencialidad física y la comunicación electrónica:

> Gifts, trust, and collective action are of course questions of status and of culture. As all social scientists know, gifts should not be thought of as arbitrary altruism [...] Yet physical setting and technology may matter too. The experimental evidence on collective action suggests that it is easier to sustain in contexts of face-to-face interaction, and 'open-outcry' trading pits provided just such a context. Free-riding and defection for private benefit from tacit 'sharing' agreements may be harder for other traders to detect when trading is conducted by telephone or electronically, and traders in those contexts may not be able to express their disapproval as effectively as they can in face-to-face interaction (MacKenzie, 2009a, pág. 24).

Siendo esto así, parecería que la no copresencialidad del entorno digital, al estar las relaciones mediadas computacionalmente, rebajaría mucho la obligación de recibir. Entre otras cosas porque, como señala Mackenzie, se podrían introducir dudas sobre los motivos del que dona. Y en efecto, aparentemente no hay nada más fácil que rechazar o ignorar un correo electrónico o cualquier forma de comunicación mediada computacionalmente. Pero esto es engañoso, porque si bien es cierto que la presencia corporal del otro no puede ser impuesta, también lo es que las comunicaciones electrónicas dejan siempre un rastro, haciéndolas presentes a lo largo del tiempo, e imponiendo de hecho una copresencialidad no sincrónica, pero más permanente. Pero sobre todo, debemos tener en cuenta que uno de

los sujetos implicados es lo que llamamos procomún digital, y en este ámbito la copresencialidad no sólo está presente sino que se multiplica. El procomún y sus repositorios están siempre presentes y, a diferencia de otros sujetos, al menos en algunos sentidos tienen tanto la capacidad como la obligación de recibir. En el capítulo 6, *Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?*, seguiremos tratando esta cuestión de la copresencialidad y su importancia en otras prácticas sociales.

5.3.2 Organizar la colaboración ¿Existe la obligación de recibir en Debian?

Debian depende del trabajo voluntario, y siempre hay tareas por hacer que podrían beneficiarse enormemente del trabajo de nuevos colaboradores. Sin embargo, en muchas ocasiones el problema no está tanto en cómo estimular la voluntad de dar o de devolver, sino en cómo articular los mecanismos que permitan *recibir*. Que esta cuestión expresa un problema real lo podemos comprobar en dos *blog posts* escritos por dos Desarrolladores de Debian:

> **How to start contributing to Debian?**
>
> I often get requests of persons who would like to contribute to Debian but who don't know where to start. Let's try to answer this question properly so that I can give out this URL the next time that I am asked.
>
> The Debian website has a page explaining how to help Debian. While it provides no less than 10 suggestions in a daunting text-only list, it's difficult to know what to do next once you picked up something that you could do (Hertzog, 2011b).

> **How YOU can help Debian!**
>
> I recently had been ask how persons usually not involved in Debian's development process can help Debian. This is a question that pops up quite often, so I thought I should write down a bit of that (Zobel-Helas, 2011).

Ambos proceden entonces a describir diferentes formas de contribuir al proyecto. Se podrían poner más ejemplos similares, pero uno de los que me parecen más claros aparece en el *post* «Cheers to audacity!» del Desarrollador de Debian Kahn Gillmor (2015):

> **Cheers to audacity!**
>
> When paultag recently announced a project to try to move debian infrastructure to python3, my first thought was how large that undertaking would likely be. [...]
>
> I'm happy to say that i also missed one of the other great benefits of paultag's audacious proposal, which is how it has engaged people who already knew about debian but who aren't yet involved. Evidence of this engagement is already visible on the py3porters-devel mailing list. But if that wasn't enough, I ran into a friend recently who told me, «Hey, I found a way to contribute to debian finally!» and pointed me to the py3-porters project. People want to contribute to the project, and are looking for ways in.
>
> So cheers to the people who propose audacious projects and make them inviting to everyone, newcomers included.

Vemos como aparece una oportunidad para gente que estaba previamente buscando una manera de contribuir: «Hey, I found a way to contribute to debian finally!», «People want to contribute to the project, and are looking for ways in».

Lo que me parece interesante destacar en estos textos es el hecho de que la motivación para contribuir ya existe previamente, el problema es cómo abrir y hacer visibles los canales que permitan esa contribución. Y éste, el de la apertura, es precisamente uno de los rasgos que definen con más intensidad al Proyecto Debian, como hemos visto en capítulos anteriores.

¿Existe pues también en Debian la tercera obligación respecto al don establecida por Mauss, la de recibir? En buena parte sí, como se observa en fenómenos como la «obligación» de atender a los informes de *bugs* o los NMU (véase el

cuadro 3.4, *NMU*). Pero sobre todo, como hemos visto, en la apertura que caracteriza al Proyecto Debian (véase la *Introducción*). Sí existe, al menos, el deseo y el intento de organizar y facilitar la recepción de colaboraciones. También existe la voluntad de ofrecer una colección de *software* libre lo más amplia posible, uno de los sentidos del lema «El Sistema Operativo Universal», lo que supone aceptar diferentes paquetes siempre que sea posible.

El requisito básico y previo para aceptar un programa en el archivo es, por supuesto, su licencia. No basta que un usuario, ni siquiera el autor original, quieran que se empaquete algo para Debian, su licencia debe permitirlo, incluyendo la redistribución. Antes de que pueda existir cualquier deber de recibir, tiene que existir la posibilidad de recibir:

> **mooch:** [Lo importante] no es si me gusta o no me gusta, es si lo puedo modificar; si lo puedo recibir sin ningún tipo de restricción; si lo puedo modificar sin ningún tipo de restricción; si lo puedo utilizar sin ningún tipo de restricción; si lo puedo redistribuir sin ningún tipo de restricción.

Esta obligación de recibir por parte de Debian sería en todo caso limitada y matizada, como la de cualquier proyecto concreto de *software* libre. La obligación fuerte en todo caso es la del procomún, puesto que nadie puede, de hecho, impedir que se produzca una aportación al mismo. Vamos a intentar observar en qué prácticas concretas en Debian se manifiesta, y hasta qué punto, esta obligación. Lo primero que hay que dejar claro es que esta obligación, entendida de manera absoluta, queda explícitamente rechazada. Así, en la pregunta 16 de las «DFSG and Software License FAQ» se establece lo siguiente:

> **Q: The program FOO is free according to the DFSG and its license; can I now demand that Debian package FOO and include it in Debian GNU/Linux?**
>
> **A:** Although Debian includes *only* free software, we do not include *all* the free software in the whole world. (Although we do include so much

> that one can understand how people might think we include it all.) What software we choose to distribute is Debian's own decision, and no one else's. In particular, we are not obligated to distribute FOO.[44]

Pero a continuación se especifican las condiciones y el proceso para que el programa pueda acabar de hecho en Debian. Un proceso que puede ser iniciado por cualquier interesado. Si se cumplen las condiciones, es bastante probable que el *software* en cuestión acabe formando parte de Debian:

> Here is what must occur for FOO to get into Debian GNU/Linux. First, it must be free, by *our* standards. Then it must be properly packaged, either by a Debian developer or by someone else. Then it must be *uploaded* to the Debian servers by a Debian developer. (This is called *sponsoring* if someone else actually did the packaging.) Then the Debian ftp masters must allow it in; they are a final screen against license issues or software integration problems. At this point the package is being distributed by Debian, but is not part of the official release. For that to occur it must be of sufficiently high quality to make it through a semi-automatic QA (Quality Assurance) process involving the Debian BTS, and the release manager must allow it to be included in the next major release.
>
> To initiate this process you can *file an RFP*.[45] See Work-Needing and Prospective Packages for details.

Son varias las razones que pueden llevar a rechazar la admisión de un determinado *software*, o a cuestionar su admisión en el archivo de Debian. La más sencilla tiene que ver con la simple utilidad. Como ejemplo, podemos citar el ITP (*Intent to Package*, Intención de Empaquetar) `nyancat`, un programa que muestra el *meme* Nyan Cat en la terminal. Este meme se hizo muy popular en 2011 (Wikipedia, s.f.[g]), y un Mantenedor de Debian mostró su intención de empaquetar

[44] En Pearlmutter (s.f.). *FOO*, junto con *BAR*, es una variable para referirse a algún elemento relacionado con la informática. En este caso, un programa.
[45] *Request for Package* o Petición de Empaquetado.

un programa que simplemente lo mostraba, sin ninguna otra utilidad, y subirlo al archivo. El *bug* relativo al ITP se puede consultar en el BTS (Bugs.debian.org, s.f.[i]), y una discusión más extensa en la lista de correo `debian-devel` (Debian-devel (2012d) y siguientes). En esta discusión se pone en cuestión precisamente la necesidad y conveniencia de empaquetar y admitir cualquier programa en la distribución.

Pero a pesar de éstas y otras preocupaciones, el mantenedor encontró rápidamente Desarrolladores que estaban a favor de que se admitiese, de tal modo que el paquete fue esponsorizado y subido al archivo de Debian, formando hoy parte de la distribución. Éste es un caso muy simple, pero muestra que, aunque de hecho no hay tal obligación fuerte de recibir, de hecho sí que hay una fuerte presunción en su favor y es fácil, cumpliendo los requerimientos, conseguir que un paquete sea aceptado.

Otro tipo de razón para rechazar la inclusión de un paquete sería su posible peligrosidad, bien para los usuarios bien para el propio proyecto. Se trata en primer lugar de evitar que se suba *malware* o *software* malintencionado, que pueda poner en peligro el ordenador, los datos o la privacidad del usuario. Este problema lo discutiremos al tratar el proceso de ingreso en el Proyecto, ya que uno de los problemas de este proceso consistiría precisamente en el posible ingreso de alguien malintencionado en el Proyecto. Los controles normales a la hora de subir un paquete se consideran suficientes en este caso. La distribución de un determinado paquete puede ser también peligrosa para el Proyecto si éste no puede distribuirlo porque su licencia no sea libre, exponiéndolo a un proceso judicial por violación del *copyright*. De nuevo, la comprobación de la licencia en NEW por parte del FTP Team ha sido suficiente para evitar estos problemas.

Más problemático es cuando alguien pretende que se suba un paquete que puede tener connotaciones ofensivas o discriminatorias. Uno de los que creó más polémica fue el intento, en 2004, de subir el paquete (ITP) `hot-babe`,[46] que mostraba el incremento de temperatura del sistema a través de imágenes en las que una mujer se iba quitando la ropa hasta que quedaba desnuda. Tras una larga polé-

[46] Véase Bugs.debian.org (s.f.[j]). Un resumen de lo que ocurrió se puede encontrar en Byfield (2005).

mica en las listas de correo de Debian, el ITP fue rechazado por los FTP Masters. Las razones alegadas para su rechazo durante la polémica fueron problemas (al principio) con la licencia, el sexismo que suponía o su posible ilegalidad en algunos países si se clasificaba como pornografía. Pero la razón del rechazo final fue simplemente que era un paquete superfluo que no aportaba nada. Por supuesto, tampoco `nyancat` lo hacía, pero en ese caso no existían las connotaciones sexistas.

Más espinoso fue lo que ocurrió en marzo de 2014 a partir de una pregunta sobre la posibilidad de empaquetar para Debian el juego «Bernd und das Rätsel um Unteralterbach», una novela visual interactiva centrada en la pedofilia y con representaciones gráficas de abusos a menores.[47] Lo que originó la polémica fue efectivamente sólo la pregunta (posiblemente con la intención de provocar, por parte de alguien sin ninguna relación previa con Debian) por la posibilidad de empaquetar el juego, puesto que ni siquiera hubo un ITP formal, ni la licencia del mismo lo permitía. Así que ni siquiera fue necesario un rechazo explícito del FTP Team. Aun así, provocó un gran discusión, en la que algunas personas entendían que si se solucionaban los problemas legales quitando determinadas imágenes y elementos del juego para eliminar los problemas legales, y se corregían los problemas con la licencia, no sería legítimo rechazar una pieza de *software* por razones morales, puesto que podría conducir a la censura. Aunque en estos dos últimos casos (mucho más en el segundo) el consenso se estableció en el rechazo a la incorporación de estos programas, y negando por tanto la obligación de admitir cualquier programa en el archivo, las discusiones mostraron también que la aceptación de los programas (por supuesto, si se cumplen los requisitos técnicos y legales) es la posición por defecto asumida de hecho, siendo necesaria la justificación en contra en cada caso particular.

Esto se ve más claramente quizás en otro hilo en la lista `debian-devel`, donde un Desarrollador plantea la cuestión de los criterios a partir de los cuales se puede rechazar un paquete (el paquete en concreto es el referido en el párrafo anterior) (Debian-project, 2014):

[47] Un pequeño resumen de la polémica y numerosos enlaces se pueden encontrar en Geekfeminism.wikia.com (s.f.). La discusión tuvo lugar en las listas `debian-games`, `debian-legal` y `debian-women`.

```
To: debian-project@lists.debian.org
Subject: clarify FTP master delegation?
From: Daniel Pocock <daniel@pocock.com.au>
Date: Tue, 11 Mar 2014 19:20:45 +0100

There is some ongoing discussion (on debian-legal) about whether the FTP
masters will accept a particular package

The FTP team wiki [1] links to a delegation email [2]

The delegation email is very light, it just says they are «Accepting and
rejecting packages that enter the NEW and byhand queues» without any reference
to the policies they should apply […]

My impression is that the type of issue currently under discussion is not
adequately specified in the FTP master delegation, it leaves the FTP masters
to do more work on something that is actually quite complicated and has
far-reaching ramifications for the project. It also means the FTP masters
are in a situation where whatever they do, some people will feel they either
did the wrong thing or some people will feel the FTP masters were wrong
to make any decision without the project having a policy on the matter.
[…]
```

1. https://wiki.debian.org/Teams/FTPMaster
2. https://lists.debian.org/debian-devel-announce/2012/10/msg00004.html

El hecho es que, dado el funcionamiento de Debian basado en la doocracia, tal y como vimos en la sección 3.2, *Toma de decisiones: de la meritocracia a la doocracia*, es efectivamente responsabilidad del FTP Team rechazar o aceptar un paquete, decidiendo si se recibe o no.

Por último, pero sin ánimo de ser exhaustivos, podemos mencionar también la posible negativa a incluir algo en el archivo por el tipo de relación que se tenga con los desarrolladores *upstream*. El siguiente correo (cuya última parte ya citamos en la sección 2.2, *Primera controversia: Sobre la diferencia entre lo técnico y lo social*, en otro contexto), escrito por un FTP Master, expone el problema, y muestra al mismo tiempo que se comparte la preocupación, expuesta en el correo anteriormente citado, por la cuestión de quién debe tomar este tipo de decisiones (Debian-project, 2009a):

From: Joerg Jaspert <joerg@debian.org>
Date: 2009/9/10
Subject: Re: Distributing software written by hostile upstream developers
To: Steve McIntyre <steve@einval.com>
Cc: debian-project@lists.debian.org

> In terms of rationale, I think it's clear that we do **not** have to
> package every piece of Free Software that is available to us. If we
> can't have a sensible relationship with the upstream developers, then
> I believe it would be better not to expose Debian and our users to the
> problems that will likely arise from packaging and distributing their
> software.

I am all in favor of being able to refuse the addition of software to Debian if we know that its upstream is a troublemaker.

I am all against having this responsibility solely with FTPMaster.

I wish/hope it will end up being with a group of DDs. That group should include FTPMaster (as we have to carry out the decision) and many DDs From all around the project. A dozen or more, diversity==win in this case. Even better if the group members term expire together with the DPLs and they have to be selected and delegated again. (Ok, that excludes FTPMaster from rotation, as we dont change that often). And every DD can send in statements in support of other DDs (or themself) to join the group **or** get another term, as well as getting them out next rotation.

Or, even harder though better: The DPL or someone appointed seeks out members each time we have an issue brought up. Not getting the same person into the group three times in a row. (Or something like it).

Yes, this makes it difficult and lotsa work. But heck, we are going to reject packages based on social and **NOT** technical standards. If that gets to be an easy thing we are doing it wrong. Especially as the perception of social standard and behaviour tends to be very different From person to person [1], so having this be done by a large group hopefully makes it better, as more viewpoints are added. [2]

[1] Imagine being the one poor guy of (select one: different nationality, color, religion, whatevercraphumanscanthinkabout) in the middle of a circle of 30 bad guys with knifes of (the opposite what you just selected). Thats a **very** different point of view. :) No, seriously - we are a very widespread project, such a group should reflect this.

[2] Sure they will have to have a way to get out of pointless discussions, but Im sure we can find/define a process for it.

Y es que, en Debian, la cuestión del don y, específicamente, la cuestión de la obligación de recibir están relacionadas con su papel como *proxy* y su relación con los desarrolladores y proyectos *upstream* y *downstream* (véase la sección 4.6.1, *Debian como* Proxy. Upstream *y* downstream). Porque Debian ocupa un lugar que la mayoría de las veces es de paso. Ya tratamos los problemas de estas relaciones en la sección 4.6, *Debian como distribución*, pero podemos mencionar aquí que de hecho son relaciones que hacen circular dones, en tanto que Debian pasa lo donado por un proyecto (los programas) a los usuarios; pero también en la otra dirección, de los usuarios a los autores *upstream*, fundamentalmente informes de fallo.

En el hilo que sigue al último mensaje citado volvemos a encontrar algunas de las cuestiones que hemos venido tratando, como el papel del Comité Técnico, la dinámica de las listas de correo y sus *flamewars*, la doocracia o la distinción entre criterios técnicos y sociales. De hecho, las últimas cuestiones mencionadas plantean un problema muy relacionado con la existencia de la obligación de recibir: quién decide de hecho sobre estas cuestiones.

Una de las hipótesis de esta investigación es precisamente que existe una conexión directa entre la doocracia y la apertura propias del Proyecto Debian (véase la sección 3.2, *Toma de decisiones: de la meritocracia a la doocracia*) y esta obligación de recibir. Si bien esta relación entre doocracia y obligación de recibir puede ser de oposición a nivel individual (en tanto un Desarrollador puede tratar de evitar las contribuciones en aquellas parcelas sobre las que tiene responsabilidad), a nivel colectivo se favorece la aceptación del trabajo que uno hace por propia iniciativa. De alguna manera, el hecho de realizarlo se convierte en un fuerte argumento para su aceptación. Respecto al funcionamiento interno del Proyecto podemos reconocer entonces una cierta tensión entre esta obligación de recibir y la concentración de poder que puede aparecer en los *core teams* (como vimos en la sección 3.2.4, *Concentración de poder y los límites internos de la doocracia*).

Durante el trabajo de campo desarrollado sobre el Proyecto Debian, encontré que una experiencia común entre los desarrolladores de Debian cuando empezaron a contribuir al proyecto es que *querían* contribuir por diferentes razones, pero que uno de los momentos decisivos era *descubrir* que podían hacerlo y

que sus contribuciones eran aceptadas, antes de ser miembros del Proyecto. Como ejemplo, el siguiente fragmento de entrevista se refiere a la situación en el *Debian Perl Group*:

> **gwolf:** En el *Perl Team*, prácticamente lo único que pedimos si una persona le indica al sistema [...] que quiere ser miembro del grupo, es que se presente. Y casi casi de inmediato alguno de nosotros le da acceso. [...] Si tú empiezas a participar en este grupo, lo que puedes hacer es modificar sobre lo que existe, pero no puedes subir los paquetes. Los sigue subiendo un Desarrollador. Si yo veo que tu modificación no cumple los criterios de calidad puedo decirte que no me sirve, que no estás haciéndolo bien. Puedo terminar revocándote el acceso. Una cosa que nos ayuda mucho es que este grupo trabaja sobre un sistema de control de versiones. Incluso si tú entras de manera maliciosa y empiezas a meter ruido, revertimos lo que hiciste y seguimos trabajando.

Hay una relación directa entre apertura y obligación de recibir. Un Desarrollador sintetiza así estas relaciones entre la doocracia, el don, y la apertura:

> **dondelelcaro:** What really matters is what the actual people here in the community are doing, I mean that they are sharing everything ... it's all open ... in fact, their openness is actually, makes it even more easy to contribute, because you don't have to wait for somebody else to unlock this piece of software.

O como vimos en otro fragmento en la sección 3.3.2, *Apertura y libertad*, «people should be free to help».

En definitiva, y volviendo a la pregunta de esta sección, si bien es cierto que no existe obligación formal de aceptar cualquier contribución, sí que podemos afirmar la existencia de una fuerte presunción en su favor. Podemos entender entonces la obligación de recibir en Debian como la obligación de articular mecanismos y dispositivos que permitan, articulen y faciliten la aportación de cualquiera

al Proyecto. Por supuesto, ni aún así se puede afirmar que esta obligación se cumple siempre o sin tensiones y controversias, y son muchos los casos en los que se disputa al respecto, como se puede ver en esta investigación.

Pero indudablemente Debian, y el *software* libre en general, han inventado y desarrollado dispositivos (tecnológicos, legales y sociales) para incluir las aportaciones de otros en el colectivo, incorporando al mismo tiempo a esos colaboradores, a los usuarios. Dispositivos como el BTS, las listas de correo, las *wikis* o los repositorios para el desarrollo colaborativo, con los que el Proyecto busca maneras de reducir las barreras para la contribución. Otros ejemplos serían el uso de la etiqueta *newcomer* (curiosamente, antes llamada justamente *gift*[48]) para señalar *bugs* especialmente adecuados para que alguien que quisiera empezar a colaborar con el Proyecto pueda hacerlo fácilmente (Armstrog, 2014); o la participación en los programas *outreachy* (Bits.debian.org, 2018; Outreachy.org, s.f.) o *Google Summer of Code* (Wiki.debian.org, s.f.[aj]). En definitiva, como decía un Desarrollador:

> **dato:** Uno de los problemas que yo creo que realmente tiene Debian es que es difícil para la gente nueva saber por dónde empezar, cómo empezar a ayudar. Se dice siempre, «pues busca un paquete y empaquétalo», cuando hay muchas áreas de Debian que necesitan ayuda. Pero yo creo que no somos lo suficientemente proactivos para ayudar a la gente a iniciarse en esas áreas. [...] Deberíamos ser más inteligentes a la hora de ayudar a la gente a incorporarse.

Por último, recordemos que la obligación de recibir, en la teoría maussiana sobre el don, se expresa más claramente en el don competitivo propio del *potlatch*. Esto es así porque está directamente relacionado con la capacidad de crear deuda, de crearle obligaciones al otro. Se relacionan así las cuestiones del don, de la jerarquía y la desigualdad, y de la doocracia. Lo veremos a continuación.

[48] El regalo o don era del usuario al Proyecto, no al revés. Esta ambigüedad fue una de las razones para el cambio de nombre. Lo que suponía esa etiqueta era que el Proyecto le hacía al usuario el don de que éste pudiera donar al Proyecto. No es un mal ejemplo para ilustrar la obligación de recibir.

5.3.3 Don y jerarquía: de nuevo la doocracia.

Volvamos a la primera obligación relativa al don, la obligación de donar. Para la tradición que va de Mauss a Godelier la hipótesis última sobre ésta es que «lo que obliga a donar es precisamente el hecho de que donar obliga» (Godelier, 1998, pág. 24). Se dona para crear una obligación, una deuda, una posición de ventaja. Como venimos viendo, donar es establecer vínculos y asociaciones, en los que también hay desigualdad y jerarquía. En cualquier caso, el acto de la donación establece una doble relación: una de solidaridad basada en la cooperación y la reciprocidad, pero también una de superioridad que se encarna a través de la deuda (Godelier, 1998, pág. 25). La del don es una buena metáfora para entender muchos aspectos de las prácticas relacionadas con el *software* libre porque no sólo ilumina la dimensión de colaboración, sino también la de producción de diferencias y jerarquías. Graeber (2011) explica esta cuestión a partir de la historia de un cazador-recolector inuit[49] que rechaza el agradecimiento por compartir la carne que ha cazado:

> «Up here we say that by gifts one makes slaves and by whips one makes dogs».
>
> «Gift» here does *not* mean something given freely, not mutual aid that we can ordinarily expect human beings to provide to one another. To thank someone suggests that he or she might *not* have acted that way, and that therefore the choice to act this way creates an obligation, a sense of debt –and hence, inferiority (Graeber, 2011, pág. 116).

Este segundo aspecto está presente en toda forma de don, pero destaca especialmente en el don agonístico o competitivo que caracteriza al *potlatch*, el don que se usa precisamente para marcar diferencias y crear jerarquías. Y es que el don como tal tiene un sentido diferente en sociedades con clases o marcadas diferencias de rango que en aquellas más caracterizadas por la igualdad, pero en todas se puede entender el don también como un elemento de competición política. Como explica Strathern: «The reciprocities and debts created by the exchange of gifts are

[49] La historia proviene del libro de Peter Freuchen *Book of the Eskimo*, pero historias muy parecidas pueden encontrarse en muchas etnografías sobre sociedades de cazadores-recolectores.

seen to comprise a form of sociality and a mode of societal integration. In Melanesia, gift exchanges regularly accompany the celebration of life-cycle events and are, most notably, instruments of political competition» (Strathern, 1990, pág. xi).

En el caso del *software* libre, como hemos ido viendo, las cosas funcionan de manera diferente, en tanto que no se dona a individuos o grupos concretos sino al procomún. Al estar mediado por ese ámbito intermedio, parecería que el don del *software* sería especial en tanto que no produciría desigualdades, diferencias de estatus entre donantes y donatarios. Aunque como vimos, la cuarta obligación del don tiene que ver con una desproporción insalvable entre lo que se recibe y lo que se puede donar, lo que a su vez nos remitía al concepto de gracia (véase la sección 5.2.4.1, *Lo imaginario y lo simbólico: lo sagrado y la cuarta obligación del don*). En cualquier caso, no hablamos de desigualdad de derechos a nivel individual.

Si sólo tenemos en cuenta la dinámica interna de Debian (y seguramente en cualquier otro caso de un proyecto concreto que se analice) esto es más problemático. En tanto que se dona no tanto al procomún sino al Proyecto en concreto, sí que se adquiere autoridad y se establece una jerarquía, según vimos al hablar de la doocracia: mientras más se colabora y se aporta, más poder se obtiene, en tanto que uno se hace responsable de más áreas o recursos del proyecto. En todo caso, si la analogía fuera adecuada, el *potlatch* se daría en el interior del Proyecto Debian, no en su relación con el ámbito del *software* libre.[50]

Según Weiner (véase por ejemplo Weiner, 1992, 18 y ss.), es el control sobre las cosas que no se donan lo que produce la diferencia, y la desigualdad: «[...] these possessions become vehicles for political autonomy. The right to control inalienable possessions can be used as the means to effect control over others» (Weiner, 1992, pág. 39). Pero como vemos, también autonomía. Sólo manteniendo esas posesiones a salvo de la apropiación por otros es posible la autonomía. Autonomía, control y jerarquía van así ligados:

> The guardianship of inalienable possessions such as these transforms difference into rank [...]

[50] Una idea parecida aplicada a Wikipedia puede encontrarse en Ortega y Rodríguez (2011).

> Out of this difference negotiated in exchange over what is not exchanged, power is generated and, under certain circumstances in which women are vital, transforms difference into hierarchy (Weiner, 1992, págs. 18-19).

Esta combinación de control y autonomía por un lado, y jerarquía y diferencia por otro, nos remite a lo que vimos sobre el concepto de doocracia (sección 3.2, *Toma de decisiones: de la meritocracia a la doocracia*). Una de las cuestiones que veíamos entonces se refería a las tensiones producidas por el control de hecho ejercido sobre determinados recursos comunes del Proyecto. Es ese control el que puede hacer surgir la autoridad:

> Through such control of inalienable possessions, authority emerges creating potential nodes of centralization that in particular circumstances sustain rank and political hierarchy. The basic dilemmas surrounding these loci of power and the challenges to the priorities of keeping possessions out of circulation appear in various local guises and forms throughout human history, revealing how the motivation for reciprocity is centered not in the gift per se, but in the authority vested in keeping inalienable possessions. Ownership of these possessions make the authentication of difference rather than the balance of equivalence the fundamental feature of exchange (Weiner, 1992, pág. 40).

Pero recordemos la diferencia fundamental en la que venimos insistiendo: las obligaciones de las que estamos tratando aquí no lo son entre individuos particulares, por lo que las dimensiones fundamentales del don en el caso del *software* libre no son la reciprocidad ni el intercambio, sino la participación en el procomún.

Si el énfasis en el prestigio, es decir en la respuesta a la pregunta de por qué se dona, lleva a la meritocracia (sistema que facilita entonces la conversión del capital simbólico y social en otros tipos de capital), el énfasis en la obligación de recibir

conduce a la doocracia. Teniendo en cuenta lo que me decía un Desarrollador en una comunicación personal:

> **gwolf:** Pero el que otros adopten mi contribución sigue en buena medida dependiendo de algo muy cercano a la meritocracia [...] un mal mantenedor crónico pierde puntos meritocráticos aunque sea muy reconocido como doócrata.

En Debian, el mecanismo de la doocracia puede funcionar porque existe un procomún. La doocracia depende del don, de la obligación de recibir. Si la propia actividad confiere autoridad y derechos, es porque de una manera u otra se cuenta con que se acepte esa actividad. El poder y la autoridad se ejercen pues doocráticamente, pero esta relación tiene un aspecto paradójico. Puesto que este hacer es en Debian un dar, un donar. Y eso incluye dar, al mismo tiempo, los medios y la posibilidad de cambiarlos. La obligación de recibir y la doocracia son, en principio, para todos. Ya hemos visto como el *software* libre explica el subtítulo del libro de Weiner, «La paradoja de guardar para (poder) donar», disolviendo tal paradoja. Por otra parte, es precisamente esto lo que se hace en la doocracia. Si no se da, si no se dona, en un sistema doocrático no es posible mantener ni retener nada. No es ya que no se retengan honores, prestigio y otras recompensas, sino que tampoco es posible conservar aquello que no se dona, que se pretende mantener sin donarlo. La única manera de que exista es precisamente donarlo.

Lo que quiero destacar aquí es que esta forma de gobierno facilita entonces enormemente, en su funcionamiento, la aportación de colaboraciones al Proyecto y la comunidad. Porque una de las consecuencias de esta forma de organización es que se establece, aunque sea tácitamente, esa obligación de recibir que es parte constituyente del *don* y condición de posibilidad de un sistema articulado en torno al don. La principal conclusión que se deriva de todo ello es que la obligación de recibir (en relación al don), la apertura de las comunidades, los repositorios y los procesos, y la doocracia son cuestiones que están ligadas entre sí. Facilitar el crecimiento del ámbito del procomún depende en buena parte de dar derechos a la gente para que contribuya y hacer esto visible.

5.4 Relaciones, valor y comunidad.

En última instancia, toda esta discusión del fenómeno del *software* libre a partir de las teorías clásicas del don sirve para entender con mayor propiedad su significado cultural y político en términos generales. Éste no se debe entender en ningún caso a partir de la contraposición entre interés económico y pura generosidad. Graeber lo resume con claridad distinguiendo la idea moderna del don de la expuesta por Mauss:

> The modern ideal of the gift, then, becomes an impossible mirror of market behavior: an act of pure generosity untrammeled by any thought of personal gain. [...]
>
> True, Mauss emphasizes that in most of the societies he was examining, there's no point in trying to distinguish between generosity and self-interest. It is we who assume the two should normally be in conflict. (This was one reason why he tended to avoid the term "gift" at all when speaking of other societies, preferring to speak of "prestations") (Graeber, 2001, págs. 161-162).

Entender, como venimos haciendo, el don del *software* como la entrada en una red de relaciones permite evitar entonces no sólo un dualismo extremo entre dones y mercancías, sino también otros que éste lleva aparejados: comunidad y sociedad, parentesco y mercado, interés y generosidad, imaginario y simbólico, social y técnico, etc. Vamos por último a señalar muy brevemente algunas consecuencias de esta perspectiva para dos cuestiones, la del valor y la de la comunidad.

Hay que tener cuidado en primer lugar de no tratar el *software* libre como si fuera fundamentalmente un depósito de valor para quien lo produce. El *software* libre se produce por el placer de hacerlo, no principalmente porque tenga valor económico. En realidad, se puede hablar de su valor económico[51] por analogía con el *software* propietario que, recordemos, tiene un momento de aparición histórica. Lo más característico de la GPL (de las licencias *copyleft* en general) puede estar precisamente en la ruptura con la asunción del sistema del *copyright* de que

[51] Véase la nota 2 a este capítulo.

el valor está en la cosa, al ponerlo en las relaciones. Hace explícito lo que ocurre con todo don:

> Mauss' ultimate point is that the "interest" involved need have nothing to do with making a profit –or even scoring a moral victory– at anyone's expense. Gifts act as away of creating social relations. They create alliances and obligations between individuals or groups who might otherwise have nothing to do with one another (Graeber, 2001, pág. 27).

El valor del *software* libre como objeto del don depende entonces del hecho de que crea y sostiene cadenas de prácticas, haciendo circular con él toda una serie de elementos heterogéneos como vimos en el capítulo 4, *El carácter híbrido del* software. Bugs, hackers, *paquetes y repositorios*. Este valor podría ser conceptualizado a partir de una teoría del valor como la de Graeber (2001, 2013), inspirada en los trabajos de Nancy Munn y Terence Turner:

> Value, I'll suggest, can best be seen in this light as the way in which actions become meaningful to the actor by being incorporated in some larger, social totality –even if in many cases the totality in question exists primarily in the actor's imagination (Graeber, 2001, pág. xii).

Graeber hace depender así el valor del significado y la acción.[52] Desarrollar esta cuestión propiamente excede con mucho el marco y las posibilidades de esta tesis, pero la sintonía con la concepción semiótica de la agencia que venimos usando es evidente. Se podría entonces caracterizar la doocracia como el sistema que reconoce que se crea valor y significado a partir de la acción, incorporando el planteamiento de Munn (citado por Graeber):

> Munn starts from a notion of activity. Value emerges in action; [...] Value, then, is the way people represent the importance of their own

[52] Para una concepción del valor que pone de manifiesto la importancia de la acción, las relaciones y el significado, así como una dimensión semiótica del valor, véase también Díaz de Rada (2007).

> actions to themselves –though Munn also notes that it we are not talking about something that could occur in isolation: in kula exchange, at least (and by extension, in any social form of value), it can only happen through that importance being recognized by someone else (Graeber, 2001, pág. 45).
>
> Rather than value being the process of public recognition itself, already suspended in social relations, it is the way people who *could* do almost anything (including, in the right circumstances, creating entirely new sorts of social relation) assess the importance of what they do, in fact, do, as they are doing it (Graeber, 2001, pág. 47).

Por otra parte, si lo sagrado y la gracia representan la fuente y depósito último del valor, el procomún (que como hemos visto toma su lugar estructural) vendrá aquí a cumplir esas funciones, incluyendo su carácter esencial para la supervivencia y la identidad del colectivo, de la comunidad.

El papel del don es también fundamental a la hora de constituir comunidades. Esto es claro en el caso del don del *software* libre y el procomún formado en torno a la cultura libre. A través de sus prácticas, el *software* libre ha servido como una de las principales fuentes de inspiración para el ámbito de la cultura libre y el procomún, pero además ha creado e implementado dispositivos para incluir a los usuarios en las comunidades de creación. Esta incorporación se realiza a través de actos que pueden ser pensados como don, al tiempo que la creación de un espacio de procomún es lo que da soporte a estas comunidades. Si hablamos específicamente de Debian, se puede decir además que ser miembro de Debian supone también mantener una determinada relación con algunas posesiones inalienables, que el Proyecto debe mantener y proteger para poder seguir donando la distribución Debian.

Lo iremos viendo a lo largo del próximo capítulo.

6 Debian como comunidad y como colectivo. ¿Quién es miembro de Debian?

> *A shape that is changing all the time but still maintains some kind of form, that is the Debian Community. It's not something you can define very easily.*
>
> — Micah, Desarrollador de Debian, Cáceres, julio de 2009.

A lo largo de todo este trabajo se ha hablado de Debian como una comunidad, al tiempo que como parte de algo más extenso, la comunidad del *software* libre. Como se explicaba en la sección 1.2, *Origen de la investigación y acceso al campo*, ésta fue de hecho una de las cuestiones que se encuentran en el origen de esta investigación. La caracterización de Debian como una comunidad está avalada por el uso de los propios miembros de Debian, que tanto en sus documentos fundamentales (especialmente en el «Contrato Social con la comunidad del *software* libre»; véase el apéndice A, *Contrato social*) como en su discurso cotidiano a través de los medios de interacción del Proyecto, así como en las entrevistas realizadas, hablan continuamente en estos términos. Por otra parte, sus formas de organización e interacción parecen remitir directamente a la teorización sobre las llamadas «comunidades virtuales».[1] Nos podemos preguntar entonces por la

[1] Para una revisión bibliográfica sobre la etnografía de las comunidades virtuales, véase Wilson y Peterson (2002). Para una visión más amplia sobre la etnografía de los medios digitales, véase Coleman (2010a).

oportunidad de representar como una comunidad a una pluralidad de sujetos que viven lejos unos de otros, que muchas veces no se conocen personalmente, y cuya interacción cara a cara tiene en cualquier caso una dimensión mucho menor que su interacción mediada por ordenador a través de Internet.

6.1 De la comunidad al colectivo sociotécnico.

Es un lugar común señalar la multiplicación de definiciones del término «comunidad» en los campos de la antropología y la sociología como un indicador de la vaguedad y la dificultad para alcanzar precisión analítica del concepto. Y sin embargo, no ha dejado de ser utilizado continuamente en las ciencias sociales por su capacidad para dar significado e iluminar las prácticas de los seres humanos cuando interactúan. Como dice Vered Amit, «the sheer proliferation of its invocations provides a backhanded testament to the continued popular saliency of this concept» (Amit, 2002, pág. 1).

La distinción clásica entre comunidad y sociedad (formulada por Tönnies, pero utilizada también por ejemplo por Durkheim o Maine, bajo otras denominaciones) presuponía una distinción evolutiva: la sociedad aparece a partir de la comunidad por un proceso de aumento de la complejidad, y por la sustitución de las relaciones totales entre individuos por relaciones parciales. Este proceso evolutivo sería también el proceso de una pérdida, la de los lazos primarios, casi «naturales», que ligarían a los miembros de un determinado grupo social por encima de las diferencias que les puedan separar. Esta visión evolutiva implicaría que comunidad y sociedad son realidades excluyentes, no complementarias. Pero hay otra manera de entender la distinción entre ambas formas:

> the transitions from mechanical to organic solidarity; from *Gemeinschaft* to *Gesellschaft*; from traditionality to rational-legality; from status to contract, which are so often treated as theories of social change, evolution and development, might rather be seen as different modalities of behaviour within any society at any given period of its history (Cohen, 1985, pág. 116).

No tendría entonces nada de extraño, en principio, la formación de comunidades en ámbitos marcados por la existencia de relaciones parciales estructuradas por relaciones de tipo societal:

> The significance of all this for our purposes is that the transformation supposedly wrought by these transitions has often been interpreted as rendering the community obsolete. The revision of this view, seeing the «contrasting» states as co-existent, enables us to see the survival, the burgeoning, the assertion of community, not as an aberration to be explained, but as a normal, expectable expression of the resilience of culture: of people's sense of self (Cohen, 1985, pág. 117).

En el campo de la presente investigación sólo se puede utilizar analíticamente el término comunidad si la entendemos como una realidad complementaria e inseparable de la sociedad. Más concretamente, se trata incluso de entender la comunidad como formación activa a partir de elementos estructurales, societales. El término «comunidad» supondría así la distinción no sólo de otras comunidades, sino sobre todo de otro tipo de relaciones. En esta línea, tanto Barth como la Escuela de Manchester han dado argumentos para considerar que las comunidades se forman a partir de relaciones e interacciones sociales complejas.[2]

Los dos elementos fundamentales y específicos en este proceso de constitución de la comunidad o lo comunitario serían, en el campo que nos ocupa, el papel del don y la constitución del procomún por una parte, y el papel de los objetos y dispositivos tecnológicos por la otra. En ambos aspectos es importante el papel del imaginario social (véanse las secciones 3.3, *Debian como público recursivo*, y 5.2.4.1, *Lo imaginario y lo simbólico: lo sagrado y la cuarta obligación del don*), en tanto son prácticas de las que depende la autocomprensión de los participantes como comunidad.[3]

[2] Véase la exposición de Banks (1996, cap. 2).

[3] Para la importancia del imaginario social en este campo véase Kelty (2005a). Para el papel de los objetos compartidos en el mantenimiento translocal del colectivo distribuido en torno a Linux, véase Ratto (2005a).

Respecto al primer elemento, y como señalamos al final del capítulo anterior, es precisamente la obligación de compartir el código fuente del *software* el principal factor que constituye la «comunidad del *software* libre» (a diferencia de otras comunidades *online*) en la imaginación de sus participantes, y la constituye además de manera horizontal. Esta horizontalidad es otro elemento típico en las diferentes teorizaciones sobre la comunidad. Como dice Coleman:

> For F/OSS hackers, it is imperative to constantly and recursively "equalize" the conditions by which other hackers can develop their skills and prove their worth to peers. As part of this equalization process, one must endow the community of hackers with resources like documentation and the fruits of one's labor (Coleman, 2005, pág. 254).

Pero además, el don es fundamental para la constitución de Debian como comunidad porque es el mecanismo fundamental de entrada y pertenencia. Se entra en la comunidad de Debian a partir del don. Ya tratamos esa cuestión en la sección 5.1.2, *¿Por qué se dona el* software *libre?* Añadiremos aquí tres fragmentos de entrevista con Desarrolladores de Debian a propósito de la relación entre la comunidad y el don:

> **amaya:** Descubro que Linux viene en distribuciones, que hay una de voluntarios, que me llama mucho la atención, que hay gente que por amor al arte se junta y se pone a hacer una cosa que funciona así de bien, porque son así de toreros y de Quijotes, porque pueden y porque quieren. Lo donan a sus semejantes. A mí me llamó mucho la atención la parte técnica pero sobre todo sobre todo la parte de comunidad, la parte de desinterés. El interés estaba puesto en que funcione bien y no en recibir nada tangible a cambio.

> **jordi:** Yo sentía gran admiración por esta gente, estaban haciendo el mejor sistema operativo del mundo para mí, y de repente te das cuenta de que puedes formar parte de ello. Y llegó el punto que pensaba que era hora de dar, no sólo de recibir.

El otro polo en esta relación de don es siempre la comunidad:

> **micah:** [A propósito de su trabajo de consultoría en una organización] One of our policies [is that] everything has to be free software. This is, the ideas of free software are in line politically and socially with our political ideas as well, so we are working in that community, we are benefiting from that community, we want to give everything we are doing back to that community. This is why I am a Debian Developer really, is for that reason. So everything we are doing is Debian-based and this takes almost all of my time.

Como vimos en el capítulo anterior, la práctica social de compartir el código fuente supone publicar (*release*, también liberar) el propio trabajo con una licencia libre. Se transforman así las estructuras de relación y se crea con ello un nuevo tipo de obligaciones y por tanto de relaciones sociales. Las comunidades que estamos investigando son pues en primer lugar comunidades mediadas por las licencias. La apelación a la comunidad que encontramos ligada a la construcción y el desarrollo del procomún no supone entonces una vuelta a lo comunitario frente a lo societal, sino más bien una puesta en cuestión de ese dualismo. En el mismo sentido se puede entender la afirmación de Kelty:

> The creation of the GNU GPL was not a return to a golden age of small-scale communities freed from the dominating structures of bureaucratic modernity, but the creation of something new out of these structures. It relied on and emphasized, not their destruction, but their stability (Kelty, 2008, pág. 208).

El papel de las licencias libres como práctica jurídica, social y técnica aparece así como fundamental, como ya vimos. Pero las licencias no son sino uno de los dispositivos y objetos técnicos que determinan las características específicas de Debian como comunidad. Llegamos así al segundo elemento esencial en la constitución de debian como comunidad. Y es que la integración y uso de los dispositivos

y objetos tecnológicos determinan algunas de las características específicas de Debian como comunidad. Como señala Kelty, esta reconfiguración afecta a la manera de imaginar la comunidad y las relaciones sociales en entornos tecnológicamente sofisticados:

> The notion of a social imaginary therefore needs some rethinking in the era of technoscientific globally networked societies. First of all, the social, as far as geeks are concerned, includes the relations among flesh-and-blood blobs of protoplasm interacting face-to-face (or face-to-screen) as well as the relations between different forms of software, hardware, networking protocols, and legal or regulatory regimes interacting in abstruse but generally precise technical ways. That is to say, geeks share an imagination of what society or sociality is: it includes the technically mediated software and networks that undergird our connectedness as much as it does any classic formulation of family, kin, nation, or corporate connection (Kelty, 2005a, pág. 201).

Para dar cuenta de este cambio, es teóricamente muy productivo complementar la caracterización de Debian como una comunidad con el uso del concepto de *colectivo sociotécnico* en el sentido de la Teoría del Actor-Red, entendiendo por tal colectivo una asociación más o menos estable de elementos heterogéneos que no incluyen sólo seres humanos, sino cualquier elemento técnico que contribuya a darle consistencia y estabilidad (véase por ejemplo Callon, 1984; Latour, 2001, 2005, 2007).

Ya hemos profundizado en el capítulo 4, *El carácter híbrido del* software. Bugs, hackers, *paquetes y repositorios*, en el papel de los dispositivos técnicos en las prácticas sociales en Debian, y en su caracterización como actores y mediadores: «ahora que hemos dejado de confundir a los no humanos con los objetos, quizá sea posible imaginar cuál es el colectivo en el que los humanos trabamos relación con ellos» (Latour, 2001, pág. 209). La cuestión es que efectivamente es imposible describir esas prácticas sociales sin introducir consideraciones sobre la interacción de los individuos también con esos objetos (híbridos y cuasi-objetos) y de éstos entre sí. O entender cómo se puede mantener un colectivo no tan grande,

pero sí geográficamente tan disperso, sin tener en cuenta todos los dispositivos no humanos que ha integrado. Por otra parte, en tanto que la puesta en marcha y el cuidado de esos medios es parte de los procesos sociales centrales de Debian, el concepto de colectivo está también ligado al de público recursivo (véase la sección 3.3, *Debian como público recursivo*). O, dicho de otra manera, el concepto de público recursivo examinado implica el de colectivo.

Y es que el *software* libre, además de proporcionar herramientas de todo tipo a diferentes colectivos, se caracteriza también por haber construido formas nuevas de incluir a los sujetos en los colectivos (véase la sección 5.3, *La obligación de recibir*), no sólo formas nuevas de comunicarse o colaborar para colectivos ya constituidos o en constitución. En este sentido, contribuye a crear un nuevo tipo de colectivo, en el sentido de que contribuye a constituirlos también como públicos recursivos.[4] Ocurre lo que Turner señala respecto a los nuevos medios:

> From a traditional point of view, new media simply offer new channels for the distribution of information. From the point of view of ANT [Actor Network Theory], however, they and their human partners collaborate in the creation of new socio-technical formations. Digital media do not just offer professionals like Romenesko a new voice; rather, they offer them the ability to build new linkages of institutions, individuals and machines (F. Turner, 2005, pág. 323).

Como en el caso del periodismo en el análisis de Turner, también en Debian se difuminan así las diferencias entre diferentes tipos de actores, entre Desarrolladores, contribuyentes diversos y usuarios. Lo que obtenemos es un colectivo, un actor-red que transforma lo que circula a través de este entramado sociotécnico, incluyendo el *software*.

En el presente capítulo nos ocuparemos por tanto también de la cuestión de la integración de humanos y no humanos en un mismo colectivo, insistiendo en el carácter procesual continuado de esta integración. Y es que en Debian, como veremos, el requisito para convertirse en un actor, así como para participar en la

[4] Sobre esta cuestión, véanse González de Requena (2012, 2016).

comunidad, es hacerlo en asociación con un dispositivo técnico (normalmente y en principio, pero no siempre, un paquete de *software*). De aquí la relevancia del uso del término «colectivo». Una asociación que abre el paso a la acción basada en la doocracia (véase la sección 3.2, *Toma de decisiones: de la meritocracia a la doocracia)*. La forma de reclutar nuevos miembros (humanos y no humanos) es así inseparable de esta forma de gobierno y toma de decisiones.

El papel *emic* de la idea de comunidad es una parte esencial de ese proceso continuo de integración y de construcción heterogéneas del colectivo. Así, en la sección 6.3, *Segunda controversia: ¿Quién es miembro de Debian?*, nos centraremos en las controversias sobre la formación del colectivo Debian. En estas controversias juega un papel decisivo la noción *emic* de comunidad, que no se entiende como un grupo dado y determinado, ya constituido. Es algo que apunta siempre más allá de sus constituyentes actuales, y que sólo aparece en sus tensiones (sobre todo, hasta dónde ha de extenderse). El colectivo y la comunidad están siempre en proceso de construcción, y de construcción mutua y simultánea. En este capítulo se sostendrá la hipótesis de que es precisamente la importancia de este concepto *emic* de comunidad (y las controversias en torno del mismo) una de las razones que llevan a la necesidad de reformar el proceso de pertenencia al Proyecto Debian. Por otra parte, en las propuestas de reforma y ampliación subyace un deseo de reconocer efectivamente las contribuciones al Proyecto. Esto nos conducirá de nuevo al reconocimiento del papel del don en la constitución de la comunidad.

El término «colectivo» se encuentra así en oposición a sociedad (Latour, 2005), pero no a comunidad. La unidad social correspondiente sería en cualquier caso el Proyecto Debian, tal como vimos en el capítulo 3, *Formas de gobierno. Doocracia, agencia y apertura*. Y es que si en cualquier momento parece estar relativamente claro quién es miembro del Proyecto Debian, es mucho más vaga (y polémica) la determinación de quién forma parte de la comunidad.

En definitiva, Debian aparece como un lugar especialmente adecuado para estudiar la formación de colectivos. Y es que, como señalamos al principio del capítulo 2, *Debian como trama sociotécnica*, y analizaremos detalladamente en éste, los dos problemas que está permanentemente tratando de resolver son la in-

tegración de nuevos individuos y la integración de dispositivos tecnológicos (fundamentalmente programas), que en numerosas ocasiones aparecen unidos.

Por último, se debe señalar que, como veremos en la sección siguiente, el uso del término comunidad y las controversias sobre quién forma parte de ella es resultado también de la distinción entre lo técnico y lo social cuya construcción describimos en el capítulo 2, *Debian como trama sociotécnica*. Así, la integración de nuevos individuos se ha podido entender como un proceso social y la integración de dispositivos tecnológicos como un proceso técnico. Aunque, como venimos diciendo y veremos detalladamente, hasta hace poco tiempo ambas cuestiones se unían de hecho en el mismo proceso social, el *New Maintainer Process*: integrarse en Debian como miembro implicaba integrar algún paquete en la distribución. Veremos más adelante como la consideración de Debian como colectivo sociotécnico contribuye también a entender esa distinción entre procesos técnicos y procesos sociales como una elaboración y no como algo originario. El intento de resolver las tensiones que todas estas cuestiones conllevan está en el origen de los procesos de reforma que estudiaremos en la sección 6.3.3, *Diversidad en Debian: otros procesos de reforma*.

Asociación con dispositivos técnicos, construcción de la distinción técnico/social, don, público recursivo, doocracia: como vemos, en las controversias sobre la formación del colectivo y la comunidad se vuelven a anudar los tópicos que venimos tratando. Pasemos ahora a considerar el sentido o sentidos *emic* de la comunidad en torno al Proyecto Debian.

6.1.1 Los sentidos de la comunidad.

Veamos pues qué sentido se le da a la comunidad en Debian, como un concepto usado por sus integrantes en sus discursos sobre sus prácticas sociales y las relaciones que establecen. Pero no lo usaremos como concepto *etic*, que mostraría la presencia de una forma específica de realidad social (incluso si considerada como esencialmente simbólica). No tendremos entonces que lidiar con las dificultades de ofrecer una perspectiva transcultural y clasificatoria sobre la comunidad.

Nos limitaremos a señalar que, en el caso de las comunidades relacionadas

con el *software* libre se ha usado en ocasiones (Edwards, 2001) el concepto de comunidades de práctica (Brown y Duguid, 2001; Lave y Wenger, 1991). Para una aplicación del concepto de comunidad epistémica (Haas, 1992) y sus instituciones, basado en el anterior, al caso concreto de Debian, véase Mateos-Garcia y Steinmueller (2008). Para el concepto, también relacionado con el de comunidades de práctica, de comunidad de producción, véase O'Mahony y Ferraro (2007), donde se aplica específicamente a Debian. Allí se afirma:

> Production communities differ from prior research on community forms in three ways that make them theoretically distinct and ripe for study. First, unlike communities of practice or occupational communities, they do not share a common employer or workplace. Second, unlike online communities, production communities must integrate individual contributions into a common pool, which can heighten interdependencies (e.g. Thompson, 1967) and the need for coordination mechanisms. Third, production communities often 'own' the output of their work and work toward collective goals outside of the scope of their employer (O'Mahony, 2003) (O'Mahony y Ferraro, 2007, pág. 7).

Además, la perspectiva *emic* es especialmente importante respecto al uso del término «comunidad» en relación a grupos de individuos que, como es el caso, se forman o articulan fundamentalmente *online*: lo importante no es saber si se adecuan o no a una concepción teórica predefinida de comunidad, sino el hecho de que ellos se imaginan a sí mismos como tal. Se trata más bien de descubrir el sentido que le dan a ese concepto y su funcionamiento. Así, no se trata de buscar rasgos que nos muestren que un conjunto de individuos que mantienen una interacción sea o no una comunidad. En su lugar, se trata de descubrir las prácticas sociales que dan origen a los significados sociales que comparten, entre ellos el de ser una comunidad o parte de ella. Como veremos, el *New Maintainer Process* será una de las principales prácticas sociales que contribuyen al sentido de la palabra comunidad. No se puede dar por supuesto un mismo sentido de lo que sea la comunidad para distintas formas de interacción y en diferentes contextos. Se intentará, en definitiva, ver cómo se articula y utiliza este concepto, siempre en la interacción de los sujetos y contextualmente. En definitiva, de lo que se trata es

de construir etnográficamente esta noción, investigando cómo los miembros de Debian dan significado, en su discurso pero también en sus prácticas, al término «comunidad», y cómo ese significado nos permite iluminar a su vez esas prácticas y esos discursos.

El hecho es que se trata de una noción *emic* central, incluso en el contexto más amplio del *software* libre, y en la relación entre éste y Debian. La expresión «comunidad del *software* libre» es ampliamente usada,[5] y el principal documento constituyente de Debian, el «Contrato social de Debian con la comunidad del *software* libre» (véase el apéndice A, *Contrato social*) muestra que se consideran una comunidad, o una parte de ella.

El significado de este término es variable a lo largo del tiempo y de los contextos, tanto para el Proyecto en su conjunto como para sus integrantes:

> **dondelelcaro:** For me it differs all the time. I mean, some people I think [are] in the community and sometimes I don't think of them. At that stage, when I was joining, for me it was a lot of the people on IRC on the #debian channels, and also the people on the mailing lists. That for me was a community, because I had never had been in a DebConf.

Veamos entonces algunas de las maneras en que distintos miembros de Debian explicitan y desarrollan el sentido del término «comunidad». En primer lugar, es una manera de caracterizar los vínculos que unen a sus miembros, ligándolos a un sentimiento de pertenencia:

> **ana:** En Debian mucha gente tiene unos intereses concretos, [...] hay gente que va por la gente, es como cualquier otra comunidad, hay gente que va por la comunidad, por la gente, por la familia, por la situación de pertenencia, hay gente que va por diversión y tiene unos planes concretos.

[5] Desde su origen en las actividades de Richard Stallman, el fundador del movimiento del *software* libre. Véase por ejemplo Stallman (2008).

> **luk:** A community is more than just the users and the Developers. Users and Developers are part of the community, but [...] it's actually the events around the Project, meeting each other in real life, in IRC, on planets, on the wiki, on the websites, everywhere actually where you get together, where you also meet users of your software, of Debian in general, but also of your packages, of what you are doing in the Project, that makes a community. So the community is actually the link you have with Debian, that you feel part of Debian, and not just like a member of something that is something on paper, but something real, something you really want to spend time on. And as a user of course that you feel that you get real help and that you know people you can ask in person or on the mailing lists to get help from. That's a community for me.

Y la vinculación personal, concreta, con individuos específicos con los que se interactúa. Éste es precisamente uno de los sentidos básicos de la comunidad en sus teorizaciones clásicas. Pero si en éstas la pérdida de este tipo de lazos era el producto del paso de la comunidad a la sociedad, aquí su aparición depende precisamente de las condiciones materiales propias de lo que esas mismas teorías llamarían sociedad. Este tipo de relación proporciona al mismo tiempo buena parte del disfrute y la diversión de la pertenencia a Debian:

> **mvz:** I have a feeling of community when I got here and meet people I have already had a conversation with on the Internet, via email or something, or just know the name from some planet blog post or something, and I enjoy to meet them, so it was fun to get to know the person behind the name. I enjoy that, it feels a bit like community [...] it is interesting to see the people, you know? I enjoy that a lot. It is also that, for example when I started working with [...] we cared a lot about the same things, and we enjoyed, it was really fun to work together, to sit there, [...] and just had lots of fun. [...] It is going out with people like yesterday, people I work with on technical stuff, we go out to drink and have fun, [...] the people are really cool people, they are really interesting and clever, and this kind of things, really interesting

> personalities as well, so I enjoy meeting these guys, and girls, yeah, I really enjoy being with the people, I think that's community to me.

Relaciones que, como vemos, desbordan afectivamente la colaboración en un proyecto tecnológico común:

> **dato:** [Preguntado por el sentido del término «comunidad» aplicado a Debian] Pues, tal como yo lo entiendo, es que si estamos aquí colaborando en un proyecto [...] hace que se hagan lazos entre esas personas de apreciación, de amistad, de respeto o de tal, entonces, que nos unen más cosas más allá del interés de que Debian sea una gran distribución. [...] queremos que Debian sea grande, buena, pero inevitablemente salen lazos que nos hacen sentir más cohesionados, que introducen cohesión entre nosotros, que hacen que, pues que nos sintamos más a gusto trabajando. [...] No es una relación entre nosotros estrictamente profesional, profesional en el sentido de hacer algo, un producto, sino que vamos más allá. [...] Yo creo que a lo mejor en *software* libre, igual depende del proyecto, pero por lo menos en Debian, es muy normal que salgan lazos bastante fuertes. Yo creo que porque a todos nos apasiona lo que hacemos, pues es más fácil a lo mejor que surjan afinidades o que nos apreciemos mucho, no lo sé ...

Estos vínculos personales no dependen exclusivamente, por otra parte, de que haya existido una interacción o encuentro personal previos. O, al menos, el interés personal por los otros. Así, son recurrentes las controversias sobre lo apropiado de enlazar *posts* sobre la vida personal en *Planet Debian* (Planet.debian.org, s.f.), la *web* donde se agregan los *blogs* de aquellos que contribuyen a Debian y quieren agregar su *blog*, o una parte del mismo. En la página *wiki* que describe su funcionamiento podemos leer (Wiki.debian.org, s.f.[ak]):

> What Can I Post On Planet?
>
> Planet Debian aims to aggregate the blog posts of people who are active in Debian and not only to aggregate the blog posts about Debian. The

> point is to provide a window into the community itself. Posts that are about Debian are a great idea and some people will choose to only syndicate «on topic» posts. But other posts are also welcome! We want to learn about the people, their life, opinions (even political) and doings.

No son raras las polémicas en torno a la cantidad de *posts* que se consideran no adecuados por ser demasiado personales, políticos, o relacionados con alguna actividad económica particular (por ejemplo, las discusiones que en noviembre de 2010 siguen a los mensajes «No general political content on Planet» (Debian-project, 2010e) o «Please draft a policy for planet.debian.org» (Debian-project, 2010f)) y que para algunos es excesiva. Pero en ellas siempre aparecen numerosas opiniones en el sentido de que es importante también el interés por aspectos más personales de la vida de los miembros de la comunidad, y el mantenimiento de ese tipo de vínculos más allá del ámbito puramente técnico. Citaremos dos ejemplos en dos mensajes de las discusiones mencionadas:

> For me, the planet is a window to the lives of the people that form up this social group. We share a technical affinity, so we tend to write technical topics, but we write about our political views - As much as we write about our hobbies, our families, our lives.
>
> For me, Debian is as much a technical project as a social one. And that's the reason I enjoy the planet (Debian-project, 2010g).

> Well. Fine. As you might know from reading http://wiki.debian.org/PlanetDebian we actually DO have a policy for content on Planet Debian.
>
> This is **intentionally** kept vague and we do not want to have it much more specific. Planet is about the people, their life and doings, their thoughts and feelings. We want as much to have about one developer's children and their adventures as we want to have the occasional rant about politics from another Maintainer. We want to have a translator write about their adventures when they are «Lost in Translation»; we also want to learn about the new company one of our fellow developers

> just founded, and what problems both face and how they solved them. We also want to learn what other things people are doing, what they like, and so on (Debian-project, 2010h).

Unos vínculos personales que desbordan pues la colaboración técnica, pero que no dejan de estar relacionados estrechamente con una forma de entender los objetivos que se intentan conseguir en común:

> **micah:** I think a community in a sense is when it has a shared vision, and goals, as working together on something. But it also means people who are not just working together on some goal, that are ... hanging out with each other, having fun together or eating together, going to the bar or whatever, so it's a little bit of a social thing. But it is the Project and the Project goals I think, and the philosophy, the Social Contract, what I think makes the Debian Community a certain shape.

> **agi:** Para mí comunidad ..., lo siento como una panda de amigos que tienen un *hobby* en común y que quieren sacarlo adelante, y que dedican un poquillo de su vida, llena de hipotecas, y de *stress*, y de no se qué, dedican un poco a una especie de proyecto utópico que es que no nos esclavice el *software* que corre en nuestros ordenadores, y que la gente pueda beneficiarse de los conocimientos. Para mí lo más básico es que el conocimiento es libre, o sea que si no se promulga, si consentimos que las grandes empresas compren el conocimiento y lo encierren en una botella y nos vendan sólo un trocito a buen precio, pues el mundo se iría al garete.

En estas citas se ve con claridad que la comunidad se encuentra en la reunión presencial, pero también igualmente en la comunicación electrónica, en el IRC y las listas de correo. Es así un término cuyo uso permite de alguna manera suavizar y superar la tensión entre esas dos formas de interacción.

Otro sentido importante que se le asigna al uso de la noción de comunidad es además, como vemos, la contraposición a otro tipo de organización que produce distribuciones de GNU/Linux, empresariales o corporativas. En este sentido es en el que se habla de Debian como una *Community Driven Distro* (distribución desarrollada por la comunidad):

> **sez:** The main thing is that it is community managed. Commitment to software freedom and technical excellence, those are good things but they are not unique to Debian. What is unique to Debian is that it is purely community managed, there is not a single company behind it.

> **agi:** [Ante la pregunta de por qué empezó a colaborar con Debian en lugar de con otra distribución]: Porque en Linux entonces distribuciones comunitarias era Debian y poco más. Slackware era comunitaria pero ya estaba de capa caída, Gentoo todavía no había aparecido, Red Hat no era un proyecto comunitario, Suse más o menos lo mismo, aunque siempre están abiertos y tal, pero no era como Debian, o sea, en Debian no había una empresa, era un grupo de personas exactamente igual que el grupo que hace el *kernel*, o Samba, o Apache, solo que era una distribución.

Es precisamente esta contraposición a los intereses empresariales o corporativos lo que relaciona el concepto de comunidad con los objetivos que no se reducen a la excelencia técnica:

> **ana:** Pero normalmente el caso es, gente que empieza con Linux, empieza colaborando en Ubuntu, [...] y al final se dan cuenta, esto ya es una opinión muy personal, que donde realmente está la libertad, y pueden elegir, pueden votar, pueden ..., lo que les motiva es Debian, y se van a Debian. Siguen teniendo relaciones con Ubuntu pero realmente ya contribuyen más en Debian [...]
>
> A la inversa hay gente que es muy activa en Debian, les ofrecen un trabajo en Ubuntu, y se van a Ubuntu. Nadie se ha ido de Debian a

> Ubuntu por el sentido comunitario. Yo estaba usando Mandrake, pero realmente la comunidad de Mandrake yo no la veía por ningún lado, y en Debian pues ves comunidad, ves comunidad. Pues que ves que la gente colabora, o sea, es que Mandrake es una empresa quieras que no, y entonces es muy difícil [que haya] comunidad ahí, es más fácil cuando tienes una comunidad.

Pero uno de los elementos más significativos sobre el uso del término comunidad en Debian es el hecho de que, aunque parece estar claro quién es miembro del Proyecto Debian (los Desarrolladores y los Mantenedores; véase la sección 6.3, *Segunda controversia: ¿Quién es miembro de Debian?*), es mucho más vaga la especificación de quién forma parte de la comunidad. En concreto, no siempre está claro si los individuos que contribuyen más informalmente al Proyecto, o incluso los usuarios forman parte de la comunidad. Por citar un ejemplo, en el libro ya citado de Krafft (que además de autor del libro es un conocido Desarrollador del Proyecto) se dice: «Another important point to consider is that members of the community (or "users" for short) [...]» (Krafft, 2005, pág. 51); «To sum this up, the community is made up mostly of users who are members by choice and contributors by conviction» (Krafft, 2005, pág. 52) dejando claro que la comunidad incluye a los usuarios. Sin embargo, en la página siguiente leemos: «Within a community of about a thousand developers, many different interpretations [...]» (Krafft, 2005, pág. 53).

O citando unas entrevistas a unos Desarrolladores, vemos que no hay acuerdo en la extensión de la comunidad. Para algunos, es más amplia que el conjunto de los miembros del Proyecto. Para otros, es menos:

> **zack:** [Ante la pregunta sobre quién es miembro de la comunidad] The most natural answer to this question is, "who is a project member", but in fact no, it is more than that. So I would say that are part of the Debian community all the people that regularly participate in the electronic life of Debian and also in the live meetings. So I don't know, every contributor which is known on IRC and which participates often, or which participates often in mailing lists. It's someone who gets

known by the others and is kind of recognized for his technical contributions but also for just being around. So I would say that just being around is not enough, and also only the technical work is not enough, you kind of need both of them to be part of the community, at least in my book, then different people may [have] different opinions.

dato: También se habla del siguiente nivel de la comunidad que no son solamente Desarrolladores sino que son los usuarios, o Desarrolladores potenciales [...] ha habido proyectos que han intentado abrir un poco Debian. [...] Están los simples usuarios que usan un producto y ya, pero están los usuarios a los que les interesa las cosas que pasan en Debian.

amaya: [Ante la pregunta de si los usuarios son miembros de la comunidad] Por supuesto. Son la principal cantera de Desarrolladores, y si no hubiera usuarios Debian no tendría sentido. Los usuarios son el motivo. Son parte de la comunidad por supuestísimo. [...] los primeros que sufren nuestros errores, que nos reportan lo que funciona y lo que no, los que agradecen las cosas. Y en el Contrato Social, el primer punto dice que todos nuestros esfuerzos van orientados al *software* libre y a nuestros usuarios, y son la comunidad, son ellos. Nosotros sólo tenemos una dirección que pone @debian.org, pero los usuarios son lo más importante que tenemos, y son parte de la comunidad, por supuesto, por supuestísimo.

Pero a veces se entiende también que este uso está sobredimensionado:

gwolf: A mí se me hace un poco exagerado como muchas veces hablan de comunidades de usuarios. Una comunidad de usuarios no necesariamente es una comunidad, es un grupo de gente, pero en este caso sí hay una cercanía a nivel personal.

> **jonas:** What is it for me? [...] I feel that I fit with Debian, and I feel family, I feel related to Debian because with Debian I have found the people that is like me. I mean, not the other way around, not me getting into Debian, more like I found them. We have always been family, I just didn't know they existed. The people that I hang out with, the people that I feel that are my friends or my relatives. My community with Debian is not all of Debian, my community in Debian is people that I feel that I can sit down and talk to. Some of the people that I sit down and talk to for many hours, now, here in Cáceres, I haven't seen since Brazil, and even maybe I don't even remember the names. It doesn't matter.

En cualquier caso, si incluso los usuarios son parte de la comunidad, es en definitiva por los principios y objetivos que se sostienen en común:

> **bubulle:** I definitely think this is a community, which includes the developers but also contributors and down, down someway, to users. Because we share some values, we are committed to free software, to our users and free software, that is one of the parts of the Social Contract. Debian is probably the only and was certainly the first project to have a Social Contract ... I think this is meant to be the cement among the community, so whatever disagreement we might have about this or that, we still have this cement, [...] and I think we all share this, and with our perception, with our own culture, we share this.

Pero en definitiva, en lo que todos están de acuerdo es en el carácter comunitario de sus prácticas en Debian, independiente de hasta donde haya que extender esa comunidad («sea cual sea la definición de comunidad»), algo que varía continuamente según los individuos y los contextos:

> **jonas:** I believe that whatever definition of community [...], Debian is a community. I mean, Debian is so big and so knitted together, so

> tightly knitted together that, yes it is a community whatever definition, I believe.

Si el uso del término está tan generalizado, la pregunta relevante es: ¿por qué los miembros de Debian se piensan como comunidad? Está claro que hay que tener en cuenta razones históricas, en tanto que el uso de expresiones como «la comunidad del *software* libre» es anterior a la existencia de Debian y es muy común, y sus miembros se consideran una parte de la misma. Sostengo que además se pueden proponer dos hipótesis complementarias.

Por una parte, el uso y el significado del término reflejan el fenómeno del don que la constituye, sería una manera de conceptualizarlo. Por otra, intentaré mostrar etnográficamente que es precisamente la importancia *emic* del concepto de comunidad y su relación con el don una de las razones que lleva a la necesidad de reformar el proceso de pertenencia al Proyecto como Desarrollador o Mantenedor, como veremos en la sección 6.3.3, *Diversidad en Debian: otros procesos de reforma*. Éste, el *New Maintainer Process*, es un proceso en muchas ocasiones largo y complejo, y que requiere un gran conocimiento técnico de la distribución. Y es un proceso que periódicamente se pone en discusión precisamente porque se buscan formas más inclusivas de pertenencia, que reconozcan más fácilmente las contribuciones al Proyecto de miembros sin el marcado perfil técnico que caracteriza al desarrollador de Debian.

Además, los componentes que hemos distinguido en este uso *emic* de la noción de comunidad (importancia de los vínculos concretos y de la familiaridad, el goce más allá de la productividad, el encuentro personal sea presencial o no, los lazos afectivos, los objetivos comunes, la vaguedad en la definición de las fronteras y su porosidad, así como la importancia de las relaciones basadas en el don y la reciprocidad) son plenamente coherentes con las teorizaciones clásicas sobre la comunidad y sus transformaciones que hemos mencionado en la sección anterior.

Podemos observar que estos componentes parecen remitir también al polo de lo social en la distinción entre lo técnico y lo social que se reconstruyó en la sección 2.2.1, *Concepciones* emic *de lo técnico y lo social*. Bien podríamos limitarnos

a señalar como caería de lleno en el campo de lo social. Pero también podemos entender que los procesos de reforma del *New Maintainer Process* que vamos a estudiar son signos (en el sentido de Kockelman) que tienen su papel en la construcción de lo que hemos caracterizado como colectivo sociotécnico, que apuntan a la puesta en cuestión y superación de esa distinción entre lo técnico y lo social. El objeto que esos signos ayudan a constituir (véase la sección 2.2.8, *Agencia y semiótica*) sería precisamente el colectivo sociotécnico, por medio de una serie de interpretantes (los conceptos *emic* de comunidad que aparecen) que contribuyen a definir la pertenencia al colectivo y sus tensiones. Si en el capítulo 2, *Debian como trama sociotécnica*, veíamos como se constituían lo técnico y lo social como objetos formando una oposición dualista, aquí vemos cómo se constituye un objeto que en la práctica supera esa distinción. La comunidad, tal como se entiende en Debian, es tanto comunidad técnica como comunidad social (dos términos usados en Debian), o más bien es una manera de superar esa dicotomía. Es un colectivo sociotécnico.

Y esta caracterización de Debian como colectivo sociotécnico nos permite también, por último, deshacer una paradoja. Vimos en la sección 2.2.1, *Concepciones* emic *de lo técnico y lo social*, que una pretensión común entre los Desarrolladores de Debian era que «no somos buenos en relaciones sociales». Pero de ser esto verdad, no se entendería cómo han conseguido mantener un ensamblaje sociotécnico tan complejo, grande y heterogéneo durante tanto tiempo. Si han tenido éxito es precisamente porque son especialistas en relaciones sociales en el seno de un colectivo sociotécnico, es decir en dar consistencia y estabilidad a las relaciones, asociando humanos y no humanos.[6] Ésta es la manera en que, en la práctica, queda superada la distinción entre lo técnico y lo social, en la construcción de colectivos sociotécnicos. De ahí también la lucidez, riqueza y reflexividad respecto a la propia acción social que he encontrado en las entrevistas realizadas.

Hagamos ahora un paréntesis para tratar algunas cuestiones relativas al papel de los dispositivos tecnológicos de comunicación, y su interacción con la presencialidad, en las prácticas sociales del Proyecto Debian y en la constitución de este colectivo sociotécnico.

[6] Sobre esta concepción de las relaciones sociales, véanse por ejemplo Callon y Latour (1981) y Latour (1992, 1996, 2005, 2007).

6.2 Interacción entre aspectos *online* y *offline*. DebConfs.

En el conjunto de mediaciones que hay que seguir para describir las acciones y prácticas sociales en Debian (véanse las secciones 6.1, *De la comunidad al colectivo sociotécnico* y 1.3, *Cuestiones de metodología*), las específicas que tienen que ver con las formas de comunicación y de interacción *online* no tienen un privilegio especial, pero sí hay que tenerlas en cuenta en su especificidad. Hemos visto también en la sección anterior que uno de los componentes de la noción *emic* de comunidad en Debian tiene que ver con la complementariedad de las interacciones entre lo *online* y lo *offline*.

Así que podemos preguntarnos por la interacción entre los aspectos de la interacción *online* y *offline*. Para ello, un contexto privilegiado es el caso de los encuentros presenciales entre miembros de Debian y especialmente de las DebConfs. No porque supongan un contexto *offline* en oposición a la comunicación puramente electrónica, sino precisamente porque muestran su interacción y su continuidad. Es aquí donde se produce con mayor fuerza la actualización de la comunidad Debian.

6.2.1 DebConfs.

Desde el año 2000, el Proyecto Debian celebra cada año una reunión, llamada DebConf, en la que buena parte de sus miembros se encuentran cara a cara durante una semana. Esta sección se basa en la observación realizada en las tres DebConfs a las que asistí: Cáceres 2009, New York 2010 y Banja-Luka 2011. Lo que sigue es una descripción general que podría referirse a cualquiera de ellas. Durante esa semana se producen una serie de charlas sobre diferentes aspectos del Proyecto, se encuentran presencialmente personas que han colaborado a través de medios electrónicos, y se celebran talleres de trabajo en los que diferentes equipos impulsan su trabajo en determinada área del Proyecto. Dado el carácter internacional del Proyecto, cada año se celebra la conferencia en un país distinto, procurando no repetir continente dos años seguidos. La asistencia es libre, y se ofrece alojamiento y alimentación a quien esté colaborando, de una u otra manera, con el Proyecto.

La semana previa, y en el mismo lugar, se celebra el *Debian Camp*, unos días más intensamente aprovechados para trabajar, en común o individualmente. Su objetivo es permitir el encuentro presencial de diferentes equipos que tengan un plan de trabajo en algún proyecto. La asistencia suele ser mucho menor, y se aprovecha también esta semana para acabar de preparar la infraestructura de la DebConf. Uno de los días de la DebConf se reserva para el *Debian Day*, un día abierto a los usuarios y a los interesados, en el que se pretende dar a conocer el Proyecto Debian. La organización de la misma, tanto la preparación a lo largo del año como los detalles de funcionamiento cotidiano, dependen del trabajo voluntario de los miembros de Debian y colaboradores que puedan y quieran participar.[7]

Esta es la gran ocasión anual en la que los miembros del Proyecto pueden verse cara a cara, y por esta razón es un momento privilegiado para participar y observar las interacciones sociales entre los miembros de Debian. Para profundizar en la importancia que los encuentros cara a cara tienen entre de los *hackers*, con una referencia especial a la conferencia internacional de Debian, es esencial Coleman (2010b):

> the conference is culturally significant because it allows hackers to collectively enact, make visible, and subsequently celebrate many elements of their quotidian technological lifeworld, whether it is by laying down cable, setting up a server, giving talks about technology, or hacking up some new source code —all of which unfolds in an emotionally charged setting (Coleman, 2010b, pág. 102).

Lo primero que me gustaría señalar es que, como vimos en la introducción, el Proyecto Debian se presenta como un lugar efectivamente abierto, tanto para la colaboración como para la investigación, y esto es cierto también para la conferencia anual. Así, se acepta con mucha naturalidad la presencia de un investigador. Tanto al inscribirme vía *web* como al registrarme a la llegada al lugar de la primera conferencia a la que asistí, y al hablar por primera vez con alguien o asistir a alguna reunión, hago siempre mención a la investigación que estoy haciendo, lo que se acepta en todos los casos como una razón *suficiente* para participar en

[7] Véanse Debconf.org (s.f.[a]) o Wiki.debconf.org (s.f.[b]).

la conferencia. Cuando el segundo día pido permiso para asistir a las reuniones del equipo de organización, me responden que no hay problema, que todo lo que ocurre allí es público.

La asistencia a estas conferencias es realmente importante y significativa para muchos de sus participantes, más allá de su utilidad instrumental para el trabajo en la distribución:

> **amaya:** De hecho la gente viene a DebConf y a veces son las únicas vacaciones que tienen en su vida laboral, y en vez de pasarlas con su familia, se traen la familia a DebConf para compaginar sus vacaciones ... hay gente que ha pasado su luna de miel en DebConf, con su pareja. Y cuando se juntan varias parejas y las mujeres pueden, las mujeres o los maridos, dicen, ah, es que aquí no venís solamente a hacer cosas técnicas, es que os lo pasáis super bien. Y se traen a la familia, y la familia repite y vienen varios años seguidos. Es para todos los públicos ...

> **jordi:** [Ante una pregunta sobre el significado del término «comunidad»] Pues estamos sentados en un sitio que lo resuelve bastante fácil. Acaba de pasar [...] que es el mantenedor de [...], y nos habremos visto unas 10 veces en nuestra vida, pero para mí es un amigo. Pues aquí nos juntamos una gente que somos muy diferentes y muy parecidos en algunas cosas, y es como una pequeña familia. Y la verdad es que entre todos nos sentimos muy a gusto, yo creo que eso es lo que hace la comunidad de la gente que hacemos Debian. [...] Y yo creo que es bastante bonito eso, yo conocí a mucha gente que realmente son una gente excepcional, tanto técnicamente como fuera, apartados del ordenador, y la verdad es que es una de las cosas que más me ha hecho crecer en Debian, las experiencias que he sacado de tener encuentros como éste, como la DebConf.

> **jonas:** Technically I think I was halfway there, and socially I wasn't there at all [...]. The big change was when I went to my first DebConf

> in Brazil 5 years ago. I was completely blown away, because of this, I'm sitting in my little space in the world, and having a lot of ideas and thoughts of, it could be possible to do this ... things like that, and it really doesn't go far, [...], then I'm gathered with a hundred people, and all of them have at least the amount of vague ideas that I have, and it completely exploded for me [...] I was so exhausted after these two weeks, and I was there for another week in Brazil [...] I could not work [...] I was so blown away.

Analizaremos ahora algunos elementos de la organización espaciotemporal de la conferencia, y mostraremos como esta reunión presencial es un espacio que muestra explícitamente la integración y la interrelación entre diferentes medios de comunicación y colaboración electrónicos, entre sí y con la presencialidad de los encuentros cara a cara.

La DebConf funciona como un espacio *fragmentario*, en el sentido de que funciona, más que como un lugar único en el que se encuentran todos los miembros de Debian, como una colección heterogénea de espacios y lugares en los que existen diferentes principios de organización. De hecho, los momentos en los que (casi) todo el mundo está presente en el mismo lugar se limitan a las ceremonias de inauguración y despedida, la cena formal que se celebra una noche en algún lugar especial, y la foto de grupo de todos los asistentes. Y sin embargo, al mismo tiempo se constituye como un tiempo y un espacio capaz de reunir y condensar las prácticas y actividades no sólo de los asistentes, sino también de los que no se encuentran físicamente presentes pero siguen sus actividades gracias a los medios electrónicos de comunicación. Así, es frecuente que se utilice este tiempo para realizar anuncios que afectan a la distribución, hacer cambios en el Proyecto y, en definitiva, marcar cambios personales y colectivos.

Los espacios principales son las salas donde se realizan las charlas y los *hacklabs* (también existen otros espacios importantes como los dormitorios, el comedor, los alrededores de las salas o los bares cercanos). Las charlas consisten normalmente en una presentación, la mayor parte de las veces dada por un Desarrollador de Debian, sobre temas que van desde el análisis de licencias a la

infraestructura del Proyecto, desde la explicación de diferentes herramientas de desarrollo a propuestas para mejorar la interacción entre los miembros del Proyecto.[8] Además, se incluyen sesiones de trabajo conocidas como *BoFs* (*Birds of Feather*), reuniones informales de gente interesada en alguna cuestión específica (Zacchiroli, 2010b). Tanto las presentaciones como los *BoFs* suelen propiciar la discusión, que en muchos casos continúa informalmente durante los días que dura la conferencia. También se utilizan las charlas para ofrecer informes de la actividad de algún grupo de trabajo de Debian, o del *Debian Project Leader*, así como para anunciar cambios en el Proyecto. Lo normal es que coincidan varias charlas y reuniones en el tiempo, por lo que se hace materialmente imposible acudir a todas ellas, incluso a la mayoría. Las charlas son grabadas y publicadas, por lo que se pueden seguir después.

Un *hacklab* es un espacio donde los *hackers* se reúnen para trabajar, individualmente o en grupo. Físicamente, consiste en una habitación acondicionada con tantas mesas y sillas como sea posible, suficientes enchufes para mantener funcionando una gran cantidad de ordenadores, y una red para acceder a Internet. Durante la DebConf, en los *hacklabs* se produce simultáneamente trabajo técnico e interacción social, comunicación cara a cara y comunicación a través de medios electrónicos, como el IRC. Por esto, se puede considerar que constituyen el corazón de la DebConf, el lugar donde mejor se pueden observar las interacciones y la vida social de Debian. Y donde se hace evidente que ésta no se puede seguir sin recurrir a los medios de comunicación electrónicos. Si estás en un *hacklab*, necesitas estar conectado al menos al IRC para ver lo que está ocurriendo allí mismo. En ellos es fácil observar a gente trabajando solos en sus ordenadores durante horas, gente colaborando en tareas comunes en torno a un ordenador, o simplemente charlando sobre cualquier tema. Independientemente de lo que esté ocurriendo en otras espacios de la conferencia, siempre hay alguien trabajando en los *hacklabs*, incluso durante toda la noche.

Si la conferencia reúne en un mismo espacio geográfico a una buena parte de los miembros y colaboradores del Proyecto, sin por ello dejar de ser una conjunción heterogénea de lugares, lo mismo ocurre con el tiempo. Si bien durante

[8] La lista completa de las charlas presentadas en las diferentes DebConfs, así como todos los recursos y la información relacionados, pueden consultarse en Debconf.org (s.f.[b]).

la semana que dura la DebConf la programación de las charlas actúa como una especie de eje temporal común, lo cierto es que los ritmos temporales de quienes la atienden no siempre coinciden. Y sobre todo, no responden a distinciones convencionales entre tiempo de trabajo y tiempo de ocio, noche y día, tiempo para la colaboración en algún proyecto común y tiempo para la socialización festiva. Aunque estas distinciones no son claras ni rígidas, existen tiempos y ocasiones reservados para potenciar el ocio y la relación personal. El *day trip*, un día reservado para una excursión o actividad conjunta; la *cheese and wine party*, en la que los participantes traen productos (especialmente vino y queso) de diferentes partes del mundo y se comparten; la cena formal en algún local; o las actividades y competiciones deportivas que se van organizando.

Respecto a la organización espaciotemporal existe una diferencia considerable entre la DebConf y el Debian Camp. Puesto que al segundo acude mucha menos gente y en él se producen menos eventos programados, unido al hecho de que normalmente acude gente más estrechamente vinculada al trabajo central en el Proyecto y que suelen conocerse más entre sí, esta semana se caracteriza por una menor complejidad de los espacios y de los ritmos. Esta diferencia se refleja también en las condiciones en las que se realiza el trabajo de campo. En la semana de la DebConf se hace más difícil tener una visión global de los acontecimientos que están teniendo lugar, y las situaciones sociales, más formalizadas pero también más diluidas en su especificidad, son más fragmentarias.

En cualquier caso, lo más característico de los tiempos y espacios propios de la DebConf es que se van autoorganizando sobre la marcha, y en esa organización juegan un papel fundamental no sólo las dinámicas de la interacción cara a cara, sino, como no podía ser de otra manera, los medios de comunicación y colaboración electrónicos. Son ellos los que dan consistencia a tiempos y espacios fragmentarios, ligándolos a otros tiempos y espacios. Son fundamentales incluso para seguir lo que sucede presencialmente en las DebConfs.

Los más importantes de estos medios son las listas de correo `debconf-discuss` y `debconf-announce`, que se utilizan intensivamente durante estas fechas, al igual que el canal de IRC `#debconf`. También tiene un papel fundamental el *wiki* preparado para la ocasión (Wiki.debconf.org, s.f.[a]), donde se va escribiendo toda la

información necesaria, desde información práctica sobre el país y la ciudad donde se celebra la conferencia, a la organización de turnos para las diferentes tareas que requieren voluntarios, como la realización de turnos en la mesa de recepción; desde la organización de competiciones deportivas sobre la marcha a la coordinación de los viajes de ida o de vuelta de la DebConf. Al ser un *wiki*, cada uno puede crear una página con la información que considere relevante, o editar las ya creadas.

Las listas de correo se utilizan de manera similar. Desde recordatorios de fechas y actividades importantes, hasta la propuesta de nuevas actividades, reuniones o talleres de trabajo, peticiones de voluntarios o preguntas sobre la localidad (geográfica en este caso), la mejor manera de estar al día de lo que está pasando en la conferencia es estar pendiente de ellas.

Pero sin duda, el medio de comunicación por excelencia en una DebConf son los canales de IRC (véase el cuadro 1.1, *IRC*). Proveen una suerte de espacio común que reúne un gran número de conversaciones simultáneas sobre los más variados temas. Se utilizan para comentar lo que ocurre en la conferencia, coordinar la organización, colaborar en algún proyecto común, consultar dudas sobre cuestiones técnicas, o simplemente para una charla casual. Además, hay un *bot*, un programa que actúa como un usuario más en el canal de IRC, que va proporcionando información sobre la llegada de los participantes a la conferencia y va anunciando los eventos. Puesto que los asistentes no suelen pasar mucho tiempo alejados de sus ordenadores, es el medio más rápido para ponerse en contacto con alguien. Incluso durante las charlas o las reuniones de trabajo es usual observar que los participantes siguen conectados y comunicándose a través de este medio. Así, es normal que los asistentes a una charla vayan comentándola a través de este medio, bien desde la misma sala donde se celebra, bien desde fuera si la están siguiendo por *streaming*.

Como ya hemos señalado, los medios de comunicación en Debian, y específicamente en el contexto de la DebConf, no se usan de manera aislada, reservando cada canal para un tipo específico de comunicación o un tema determinado. Lo que da su carácter especial a la comunicación dentro del Proyecto Debian es precisamente la unión y la superposición, en el mismo tiempo e incluso muchas veces en el mismo lugar, de diferentes canales.

Como ejemplo podemos considerar una de las reuniones de trabajo a las que asistí en la DebConf celebrada en Cáceres. Se planteó como un *BoF* (Penta.debconf.org, s.f.[b]), una reunión de coordinación para los integrantes de Debian interesados en la *internationalization*[9] de la distribución, su adaptación para poder ser adaptada a otros idiomas y otras convenciones locales. La sesión de trabajo se desarrolló en una de las salas destinadas a la celebración de charlas, y fue ofrecida por *streaming* para que fuera accesible desde cualquier lugar con una conexión a Internet. También se podía seguir por el canal de IRC `#debconf-lower`, es decir, el dedicado a los acontecimientos celebrados en la *lower talkroom*, una de las salas reservadas para las charlas. Al inicio de la reunión se utiliza el canal de IRC para comprobar que efectivamente los que están fuera pueden ver y oír el vídeo. Así, aunque acudieron 17 personas a la sala, en el momento de su inicio había 67 personas conectadas al canal de IRC. Esta disposición permitía que se participase en la reunión desde cualquier parte de la conferencia, o desde cualquier parte del mundo. Uno de los asistentes presenciales va informando a través de IRC de los temas que se van tratando en la sala, y comunica las aportaciones que se van realizando por este medio, de tal modo que la participación de aquéllos que no se encuentran en la sala sea efectiva. Una consecuencia adicional es que, de esta manera, el *log* del canal puede servir al mismo tiempo como «acta» de la reunión, acta en cuya elaboración cualquiera puede participar. Posteriormente, se publicará un informe de estas sesiones de trabajo en la lista de correo dedicada a la «internacionalización» y la «localización» de Debian.

En definitiva, podemos considerar la DebConf como la situación que ilustra con mayor claridad la constitución, la realización de la comunidad Debian como colectivo sociotécnico. Lo fundamental es la densidad de la trama de relaciones y asociaciones, que da cuerpo a y actualiza la noción *emic* de comunidad.

6.2.2 Sobre la comunicación a través de medios digitales.

En definitiva, y cómo planteábamos antes, sería engañoso entender las reuniones presenciales como opuestas a la comunicación *online*. Ambos contextos, o mejor simplemente medios de comunicación, están inextricablemente relacionados,

[9] Véase la sección 1.2, *Origen de la investigación y acceso al campo*.

y no sólo en el caso especial de la DebConf. Esta es la conferencia que intenta reunir cada año a todos los miembros de la comunidad, pero cada vez más se intenta realizar reuniones cara a cara entre los miembros de diferentes equipos. Por ejemplo, los *sprints* o reuniones de desarrolladores: «Sprints, or developer gatherings, have proven to be very effective in get important work done and strengthen the Debian community» (Wiki.debian.org, s.f.[al]).[10] En palabras del DPL en 2011, «Enabling volunteer developers to meet and hack together in person is possibly *the* most valuable way of using donated money» (Zacchiroli, 2011a),[11] lo que da la medida de la importancia de las reuniones presenciales *face-to-face*.

Además, los efectos de estos encuentros cara a cara van mucho más allá del periodo en el que tienen lugar, dejándose sentir a lo largo de todo el año y contribuyendo a configurar la experiencia de la propia comunicación *online*, como se puede observar en los siguientes fragmentos de entrevista:

> **dato:** Yo creo que verse en persona es crítico, siempre digo que es importante hacer reuniones cara a cara, hay más ancho de banda, es más rápido. Se suele decir que cuando conoces a alguien luego la comunicación electrónica es más fluida, desaparecen ambigüedades, es más difícil insultar. Hasta que no conoces a la persona cara a cara es un nombre en una lista. Una tercera cosa que se dice es que cuando pones a una panda de personas, de *frikis* o como los quieras llamar en un sitio unos días, empiezan a pasar cosas mágicas, de repente salen grandes ideas. [...] Cuando las personas están juntas tiempo pasan cosas que no pasarían por medios electrónicos.

> **ana:** Si colaboras activamente en Debian es algo que tiene un impacto muy grande en tu vida. [...] Hay muchas amistades muy muy buenas y también odios viscerales. Yo lo del odio no lo entiendo, pero lo de las amistades sí. [...] A partir de un día coincides en un *meeting*, hablas más y ya estableces una [...] Le pones una cara al *mail*. Es importante porque el que está escribiendo un *mail* es una persona con sentimien-

[10] Véase también Wiki.debian.org (s.f.[am]).
[11] Véase también Debian-devel-announce (2010b).

> tos, eso todo el mundo no lo ve. Cuando estás hablando de una cosa por IRC, la única información que te llega de la persona son las ideas, y cuando estás hablando también te llegan gestos. Por IRC no se ve a la gente dubitativa. Y luego aumenta muchísimo el sentido de comunidad.

La interacción es entonces claramente distinta, siendo la presencial más rica, más auténtica, lo que facilita también la interacción:

> **bubulle:** [Hablando de la interacción cara a cara, festiva, que se produce en la DebConf] I think that is a good picture of people in Debian, being more than only technical enthusiasts working on technical stuff. [...] Interaction by electronic media is fake in some way, and having the opportunity to meet people in real life makes things much much smoother.

Pero es necesario tener en cuenta que las dificultades no dependen sólo de las características de la comunicación computacionalmente mediada,[12] sino del simple hecho de que no se conocen previamente. Una vez que se han conocido cara a cara, la comunicación por medios electrónicos puede ser también mucho más fluida:

> **agi:** Se puede conseguir que Internet sea un medio de comunicación muy útil y no tan frío como se quiere ver. Se pueden tener conversaciones de cualquier índole usando Internet, pero cuando no conoces a alguien pues siempre tienes más reparo a la hora de decirle alguna cosilla por si le sienta mal. Cuando no conoces a alguien *face to face* tienes que ser más prudente, más recatado, y luego ya cuando ves que es una persona normal, como tú, de carne y hueso, que sabe qué es un chiste y que le puedes tomar el pelo, que un día te lo encuentras y te

[12] Aquí no estamos considerando las dificultades específicas de la comunicación a través de listas de correo con un gran volumen, que plantean sus propios problemas. Éste es un tema recurrente en el seno del Proyecto Debian. Véase por ejemplo la discusión que sigue a Debian-project (2009c).

> da un abrazo ... ver a alguien en persona sí cambia, lo facilita mucho, lo hace más natural.

> **mvz:** Email is a very difficult medium to ... There's lots of information missing you have in a direct contact, so sometimes I am not sure what people mean or how they feel about things. [...] I, seeing how they have fun or something like that, will change the way I perceive stuff I read from them, on IRC or on the mailing lists. I think I will in a way enjoy it more, but also it will be easier to work together, to do stuff. Because I sort of, I don't know them well, you know?, but I have some impression of the person behind the other computer.

Y los lugares más apropiado para producir estos encuentros presenciales son, como hemos visto, las DebConfs. Este efecto es una de la principales razones que hacen que estas conferencias sean tan importantes para la vida del Proyecto Debian. En palabras de unos Desarrolladores:

> **micah:** [preguntado por la importancia de la DebConf] Well, the most important thing is seeing lots of these people that I'm working with online. And hanging out with them and talking with them, not necessarily about the work that we need to do, but just becoming friends or reestablishing social connections. [...] I have a lot of friends in the Debian community because I've come to several DebConfs. And maintaining these friendships online is difficult. I still think someone is my friend but I can't have interesting conversations in person with them, [...] leave DebConf and then people become IRC nicks and it's just text and very one-dimension. You remember what they look like, and you remember who you get along with more than other people, and it helps mediate the online communication more. You know that someone is making a joke in an email, because you know that they are always joking with you so, I'm not going to read this email like they are being mean to me or something. So it adds some of this dimension to communication. Those things are one of the most important things for me being here.

> **marga:** Fuimos [a la DebConf de 2004 en Brasil] y fue como un antes y un después, porque ahí entramos en contacto con la comunidad y eso cambia muchísimo. El que dejen de ser nombres desconocidos, perdidos en Internet, que pasen a ser un nombre y una cara. [...] Después de conocerlos personalmente, todo tiene otro color, como que la comunicación es totalmente distinta después de haber conocido a la gente personalmente.

Para un participante habitual, la conferencia puede ser igualmente fundamental incluso aunque no pueda asistir presencialmente a la DebConf. Al acabar la DebConf 13, un Desarrollador escribía en su blog (Wolf, 2013):

> As I slowly read my good friends wishing each other a good trip, telling they got home safely, and the IRC channels form thick drops of a bitter-sweet etheral substance, I cannot help feeling DebConf13 is over — For me as well, from the distance. Many friends gave me warm greetings, and without being there, gave me that beautiful feeling of real community that Debian has given me for ten years already, since I met in real-life many of its developers at DebConf3 in Oslo. And –yes, I have stated this far too many times– I have attended every DebConf since (and worked organizing most of them). [...]
>
> This year, I was unable to attend due to calendar clashes. Even so, without the stress that organizers have, and thanks to the great work of the always-loved Video Team, I think I was able to be present at more sessions than at in any of the last few years. [...] And, as always, I was able to follow many very interesting talks and take part of a couple interesting meetings/BoF sessions. I still have a bit of catchup, partly due to the timezone difference (I was only at one of the sessions during the Swiss morning, at 02:30 local time, the pkg-ruby-extras team BoF).

Si, como planteábamos en la sección 1.3, *Cuestiones de metodología*, partimos entonces de la experiencia etnográfica y de la experiencia que tienen los

miembros de Debian, en la descripción del lugar en el que se desarrollan los acontecimientos en la comunidad Debian, hay que partir de la imposibilidad de sostener un dualismo entre formas de interacción *online* y *offline*, y por lo tanto, entre formas de socialidad «virtuales» y presenciales. Así, cuando dos personas están hablando por IRC sentados en la misma habitación, posiblemente quedando para tomar algo cinco minutos después, o comentando el ambiente del lugar en el que ambos se encuentran físicamente presentes, ¿estamos ante una comunicación *online* u *offlline*? Llegamos así a la imposibilidad de conceptualizar el Proyecto Debian o la «comunidad» Debian como una «comunidad virtual», definida por un tipo de espacio virtual bien delimitado y opuesto a un espacio que sería el propiamente «real».

Es más productivo partir de las conexiones entre diferentes espacios y prácticas sociales. En Debian existen diferentes espacios en los que tienen lugar las prácticas e interacciones sociales: encuentros cara a cara, IRC, listas de correo, *blogs*, BTS, sistemas de control de versiones, o *wikis*, la mayoría de ellos del tipo que tradicionalmente se han definido como «virtuales», no dependientes de la copresencialidad física. Y por ello mismo, se han entendido muchas veces como un dominio propio, un espacio («virtual» o «ciberespacio») liberado de las constricciones de lo que se supone como verdaderamente «real». Pero estos múltiples espacios «virtuales» no existen de manera paralela entre sí, por decirlo así, uno al lado del otro sin entrecruzarse. Se conectan unos con otros (sin presuponer jerarquías *a priori* entre ellos), en la misma trama, mediante los diferentes dispositivos tecnológicos. Los mismos acontecimientos circulan entre estos diferentes espacios, transitando así de uno a otro, dando así cuerpo a la trama o colectivo sociotécnico.

Así, el espacio social de Debian no puede ser adecuadamente entendido a partir de las teorizaciones al uso sobre los «espacios virtuales» o «comunidades virtuales» (por ejemplo Holmes, 1997; Jones, 1995, 1998; Rheingold, 2000; Smith y Kollock, 1999). De alguna manera, la interconexión de lo que podríamos haber llamado «espacios virtuales» reintroduce (si es que alguna vez estuvieron fuera) en la interacción social computacionalmente mediada las constricciones de toda interacción social situada. Existen toda una serie de interacciones, confluencias y solapamientos entre las diferentes formas de comunicación, incluidas las que

se producen cara a cara. Del mismo modo en que la comunicación verbal y la no verbal no suponen dos ámbitos de comunicación diferenciados, sino que ambos forman parte inseparable del mismo acto comunicativo. Volvemos al concepto de lo social basado en las conexiones y asociaciones más que en la idea de contexto social como marco, como vimos en la sección 2.2.7, *Sentidos de lo social y lo técnico*. Existe por lo tanto una cierta redundancia de la información transmitida, que contribuye a la creación de ese espacio social complejo. En cierto modo, contribuye a la superación de las limitaciones que supone la comunicación *online*, caracterizada tradicionalmente como de *banda estrecha*.

Veamos ahora algunos ejemplos e interpretaciones de esta experiencia:

> **ana:** Verás, es que muchas veces hay un correo y nadie le responde, pero a lo mejor ese correo ha generado muchísimo *feedback* en IRC, a lo mejor alguien ha posteado "He descubierto que la mejor forma de crear paquetes rpm es usar alien" y a ese tío a lo mejor lo acribillan, pero lo acribillan en el IRC. [...] El IRC es un poco como la tele, cuando estás viendo las listas de correo, vas viendo los programas, lo que pasa es que tú puedes poner tu propio programa, hay muchísimos canales, y luego el IRC es como, estás sentada con tu madre al lado, y estás comentando, oye pues la película ésta yo estoy convencida de que éste va a matar a éste o que éste es el malo, es igual, ...

Ambas formas de comunicación tienen sus pros y sus contras, que dependen también de las costumbres de los miembros y equipos de Debian:

> **jordi:** Depende de la gente y del equipo, y de los hábitos que tenga la gente del equipo, pero en Debian están evidentemente las listas de correo, las listas de correo son centrales, pero también el IRC es muy importante, porque hay muchas discusiones a tres bandas, que si los tres tienen la costumbre de estar en el IRC permanentemente conectados, como hacemos muchísimos, es mucho más fácil y dinámico. En 10 minutos a lo mejor han resuelto una duda entre cuatro o cinco que

> están en el mismo canal, en el canal de desarrollo de Debian, que a lo mejor hubiera costado dos días en la lista de correo. Claro, tiene sus pros y sus contras, la gente que no estaba en ese momento mirando el canal no participa en la conversación y se han tomado decisiones, y se quejan de que esas cosas se deben comentar en la lista de correo que es el sitio. Pero la inmediatez a veces hace que valga la pena el tema del IRC. En Debian hay dos [...] , el tema de la inmediatez del IRC, y las listas de correo que ofrecerán a todo el mundo la oportunidad de debatir y de participar, pero que no están en tiempo real y dinámico.

Y no son intercambiables:

> **jonas:** I'ss on IRC where Debian happens. The social network is on the chat network. The people hanging out there are the ones that want to talk, and I didn't do chat, so I didn't know these people. [...] it's like the casual talking, discussion of, just saying what I am doing right now, ... It's similar to when you are in the same room, that you actually talk to a neighbor, to people next to you, even if they do something else, ... exchanging your frustrations or [...] things like that. And that's different from the mailing lists, where you have a topic, you raise topics and then you discuss the topic and you exchange ideas and you have opinions.

Podemos señalar a partir de las últimas citas que la comunicación a través del correo electrónico y las listas de correo suponen una construcción temporal distinta a la comunicación a través de IRC, en tanto la primera se caracteriza por la asincronicidad y la segunda por su sincronicidad. Pero también suponen una organización espacial distinta. Así, los archivos de las listas de correo tienen una dimensión (ordenación) temporal, por fecha de envío, pero también espacial, una ordenación por hilos y por respuestas. Lo mismo ocurre con el IRC. Si en la dimensión temporal se caracteriza por la posibilidad de mantener conversaciones en «tiempo real», en la dimensión espacial se caracteriza por su organización en salas o canales que reúnen localmente a un grupo de personas, y por la posibilidad de mantener una conversación en un lugar privado. No es necesario insistir en

que estos espacios y tiempos no están desconectados, sino que se interrelacionan continuamente con otros lugares y momentos. Como muestran estas interpretaciones, tanto el IRC como las listas de correo son esenciales para seguir la vida social del Proyecto, y lo son no por separado, sino en su interacción.

En cualquier caso, el espacio social de Debian constituye una trama en tanto se compone de una multiplicidad de lugares concretos y espacios parciales interconectados, y a través de los cuales se asocian los diferentes actores. Comprender la construcción de estos espacios es esencial para entender el sentido de lo que sea la «comunidad Debian». Así lo expresa un Desarrollador de Debian:

> **micah:** I don't know, it's an interesting question because it's a virtual community, and the space that the people who are involved in the community is not very clearly defined. There are mailing lists, and there is IRC, those are probably the most clearly defined spaces where the community gathers to talk socially, or work out problems, or work together on technological projects or non-technological Debian related things. But usually I think of our community, who is together in a space, and Debian space is very weirdly defined [...] I in the last year have not paid any attention to the mailing lists, for example, because it is too much information for me. But I still feel part of the community, maybe because I'm always hanging out on IRC, I don't know, or because I come to DebConf and build these social connections, and those last when I'm away from people. [It is] hard to define what it means community, but I think there is a kind of structure that maybe isn't the best structure or has some cracks in it, [...] but there is some structure, or a shape that is Debian, and the people inside of that shape are kind of part of the community. Sometimes those people are just the users, and sometimes it's the Debian Developers, or sometimes it's the people who are here, and it changes all the time. Like there's a different community here at the DebConf, than is on the `#debian-devel` IRC channel. Only certain people are coming there all the time, so I think it's very interesting than it can be ..., a shape that is changing all the time but still maintains some kind of form, that is the Debian Community. It's not

something you can define very easily.

Hemos visto en los capítulos anteriores (por ejemplo, en la sección 2.1, *Bug #573745*) cómo los diferentes medios aparecen interrelacionados. Los acontecimientos, las interacciones y las controversias circulan entre diferentes espacios y canales. Por ejemplo en *Planet Debian* (Planet.debian.org, s.f.), la plataforma que agrega los *posts* de los miembros y colaboradores del Proyecto. Así, el mero hecho de su agregación en una plataforma hace que aparezcan relacionados en el mismo espacio. De la misma manera, es posible seguir en gran medida el desarrollo de una DebConf, por ejemplo, a través de los *blogs* de los asistentes, a través de *Planet Debian*. Otros episodios de esos mismos acontecimientos ocurren en IRC, las listas de correos, el BTS, *wikis* o repositorios de *software*. Quizás lo más significativo sea el hecho de la integración de la discusión en una lista de correo pública con el seguimiento de informe de *bug* en el BTS, mostrando así la estrecha interrelación entre los medios de comunicación y los medios técnicos de colaboración.

En definitiva, la importancia de los dispositivos tecnológicos y la interacción entre aspectos *online* y *offline* que nos ha hecho pasar de la concepción *emic* de la comunidad a la noción de trama y colectivo sociotécnico puede considerarse como el punto de partida para el planteamiento de la siguiente controversia: si Debian es un colectivo sociotécnico, ¿qué actores forman parte de ese colectivo?

6.3 Segunda controversia: ¿Quién es miembro de Debian?

Vamos a partir, para tratar esta cuestión, de una controversia que surgió justo cuando comenzaba mi trabajo de campo sobre Debian. Tratar de entenderla fue uno de los primeros hilos que seguí para la comprensión de la vida social en Debian. Esta controversia afectaba directamente a los procesos de entrada de los colaboradores en Debian. El hecho de que en torno a esta cuestión se anuden muchos de los temas que hemos ido desarrollando, hace que su discusión sea oportuna como última parte de esta tesis doctoral.

El origen de esta controversia está en el siguiente correo electrónico (debido

a su extensión, se cita sólo parte del mismo), a la lista `debian-devel-announce`, del 22 de octubre de 2008 (Debian-devel-announce, 2008):

> To: debian-devel-announce@lists.debian.org
> Subject: Developer Status
> From: Joerg Jaspert <joerg@ganneff.de>
> Date: Wed, 22 Oct 2008 23:33:28 +0200
>
> **Developer Status**
> **Summary of this post**
> Discussions in the past have made it clear that the current definition of «Debian Developer» (AKA someone who is a member of the Debian project) should be modified and made more flexible. There have been attempts in the past to do something similar, notably Debian Maintainers (DM) [GR-DM], and to some extent debian-community.org [D-C], but these have only addressed parts of the whole issue.
>
> We plan to integrate DM more closely into the NM process/system while keeping the spirit of easing entry into Debian for newcomers. At the same time we add a separate track for less-technical contributors. […]
>
> Currently becoming a Debian Developer means passing through all of the New Maintainer process. People that passed this get the @debian.org mail-forwarding, an account on all (developer-accessible) Debian machines, voting and upload rights. It is a process that requires work from prospective Developers, and depending on their available time and the effort put into it, it can take a bit of time.
>
> Some time ago a few Developers thus went and pushed forward the «Debian Maintainer» status. DM allows newcomers to upload their packages relatively early, without having to go through the «full» NM process. So far it has worked quite well for the people involved, but the way it was instantiated outside of most existing structures has always made other groups in Debian uncomfortable. The ftp-masters have to deal with the technical implementation that does not fit well with the rest of the archive, and the account and keyring managers would like to remain the authoritative source for «who is in Debian».
>
> Debian is about developing a free operating system, but there’s more in an operating system than just software and packages. If we want translators, documentation writers, artists, free software advocates, et al. to get endorsed by the project and feel proud for it, we need some way to acknowledge that. This is where our proposal comes in.
>
> Now let us describe the way the account status is meant to be handled in

```
future.

A new user can start out in two ways depending on their personal preference.
The first is the non-technical way:

Debian Contributor
A DC is someone that has a strong relation with Debian through the work
they are doing for/around Debian. Possible examples are translators and
documentation writers.
[…]

The second way is the technical one:

Debian Maintainer
A DM has the same strong relation with Debian a DC has, but additionally
wants to maintain a limited set of packages without the help of a sponsor.
[…]

After the 6 months time in Debian Contributor/Maintainer are passed, applicants
can apply to get Debian Developer status. There are now 2 different «classes»
of DD status available, one with and one without upload rights. To not
add confusion we selected to name them «Debian member» (no upload rights)
and «Debian Developer» (upload rights). Both are project members, i.e.
with voting and all other constitutional rights, the term «classes» does
not indicate any kind of «first» or «second» level membership.
```

Como vemos en el primer párrafo, la controversia se produce en torno a la definición de Desarrollador de Debian o Miembro de Debian. Antes de analizar los cambios que se proponen aquí y las discusiones que siguieron, es necesario explicar cómo era el *New Maintainer Process* (también conocido como *New Member Process*, véase más adelante) o NMP, el proceso existente en ese momento para llegar a ser parte de Debian, y en qué consistía precisamente esa pertenencia.

6.3.1 El *New Maintainer Process*.

El NMP se formalizó en 1999. Para la historia de este proceso anterior a mi trabajo de campo me basaré fundamentalmente en Coleman (2013), Coleman y Hill (2005) y Wallach *et al.* (2005); también pueden consultarse Sadowski *et al.* (2008) y Mateos-Garcia y Steinmueller (2008). Durante los primeros años de existencia del Proyecto, el proceso de entrada era muy informal, y muchas veces bastaba enviar un correo electrónico o un encuentro personal para ser admitido.

En 1997 se establece un proceso con procedimientos más formales, y de manera incipiente aparecen ya algunos de los rasgos principales del NMP posterior: comprobación de la identidad, y comprobación de la competencia técnica y del compromiso con los principios que regían Debian, todo ello normalmente a través de una llamada telefónica (Wallach *et al.* 2005, pág. 3).

Pero durante los años 1998 y 1999, se produce una crisis de crecimiento (Coleman, 2013) o crisis ética (Zacchiroli, 2012b, 77 ss.), en la que muchos miembros de Debian sienten que se admitían nuevos miembros de forma muy rápida, sin tiempo de integrarlos propiamente en el Proyecto, y sin asegurarse suficientemente de su capacitación técnica o de su compromiso con los principios del mismo. La queja principal parecía ser la voluntad por parte de un número significativo de nuevos Desarrolladores de incluir en la distribución *software* que no cumplía las DFSG (véase el apéndice B, *DFSG*), pero que haría más fácil el uso generalizado de Debian. Esta situación provocó incluso que se llegase a parar la admisión de nuevos miembros hasta que se solucionaran esos problemas (Debian-devel-announce, 1999).

Así que en octubre de 1999 se introdujo formalmente el *New Maintainer Process*, en un correo a la lista `debian-project` (Debian-project, 1999).[13] En este correo se establece una primera versión del proceso que enseguida analizaremos, y se insiste especialmente en la necesidad de asegurar el compromiso con los principios del *software* libre. Ya desde este primer momento se menciona la necesidad de tener en cuenta colaboraciones más allá del empaquetado, lo que más tarde se convertirá en una cuestión central. En este momento se crean también el sitio «Debian New Members» (Nm.debian.org, s.f.[a]), donde se lleva a cabo todo el proceso, y el *New Maintainer Team*, encargado de llevarlo a cabo. El proceso se fue formalizando y revisando levemente durante los años siguientes.

El cambio más importante fue la introducción de la figura del *Debian Maintainer* en 2007, mediante una Resolución General. El problema que algunos percibían en el NMP era que, mientras que sólo los Desarrolladores de Debian podían subir paquetes al archivo de la distribución, hay colaboradores que sólo quieren mantener algún paquete en el mismo, y el NMP es demasiado largo y complejo

[13] Véase también Debian-devel-announce (1999).

para algunos de ellos, o simplemente no quieren pasar por él. La solución a esta situación era recurrir a un *sponsor*, un Desarrollador de Debian que sube el paquete al archivo en nombre de la persona que efectivamente lo mantiene. Pero esta solución tiene el inconveniente de que hace más difícil y tedioso el trabajo de la persona que mantiene el paquete, al tiempo que aumenta la carga de trabajo del *sponsor*.

Aquí podemos señalar una paradoja. Veremos más adelante que el NMP es un proceso tedioso y en gran medida inútil para quien quiere ser Desarrollador (Miembro de Debian), pero no mantener paquetes, debido a que profundiza mucho en esta cuestión. Pero curiosamente, también aparece como tedioso e inútil para quien quiere hacer *justo lo contrario*, mantener paquetes exclusivamente sin participar en la vida de la comunidad Debian. En palabras de un Desarrollador de Debian:

> **luciano:** Notamos que mucha gente lo único que quería hacer era mantener paquetes. Por ejemplo no les interesaba votar. En vez de pasar por este proceso tan tedioso. Por ejemplo, el P&P [Philosophy and Procedures], analizar licencias, tenías que leerte licencias de esas que son bien enrevesadas y decir si eran libres o no. Y la realidad es que el trabajo del mantenedor dista bastante de eso, o sea una vez que leíste la propia licencia [del paquete que mantienes] ya es suficiente. Entonces lo que se hizo fue eso, si vos tenés un paquete la primera vez necesitas conseguirte un *sponsor*, pero después ya mantenlo solo. Era lo que notábamos, que mucha gente buscaba el ser Debian Developer y que sin embargo se frustraba demasiado en el proceso. Porque era largo, de meses.

Dado que esta figura introduce una ambigüedad terminológica respecto al significado de «mantenedor» (un tipo de miembro, pero también designa la función de empaquetado la realice quien la realice) que se irá viendo progresivamente complicada y que se tratará de resolver en las diferentes reformas, introducimos los cuadros 6.1, *Terminología sobre miembros, mantenedores y desarrolladores*,

y 6.2, *Terminología sobre miembros, mantenedores y desarrolladores, parte 2*, para tratar de aclarar los términos usados.

En 2006, el DPL recién elegido (Towns, 2006) y un miembro del Front Desk (Berg, 2006), que más tarde sería elegido como DAM (véase el cuadro 6.1, *Terminología sobre miembros, mantenedores y desarrolladores*), propusieron una figura intermedia, la de *Debian Maintainer* (Debian-newmaint, 2006):

> Here's my proposal:
>
> Introduce an intermediate role «Debian Maintainer» (DM). Summary: is allowed to upload packages already in the archive by himself. Needs sponsoring for new packages, no vote rights. Can either proceed to become DD or stay a DM.

El Mantenedor Debian podría entonces subir por sí solo paquetes al archivo de Debian, pero sólo aquéllos en los que se le permita específicamente (usualmente, que haya mantenido previamente a través de un *sponsor*). Por el contrario, un Desarrollador de Debian puede, técnicamente, subir cualquier paquete al archivo. El Mantenedor de Debian no puede votar en las Resoluciones Generales, y tiene un acceso restringido a los recursos, servicios y máquinas del Proyecto. Todo esto supone además que su clave criptográfica ha de almacenarse en un *keyring* diferente al de los Desarrolladores de Debian (véase la sección 6.4.1, *Identidad y claves GPG*). Desde el principio, la propuesta se entiende bien como un paso intermedio hacia la posición completa de Desarrollador Debian, bien como una posición definitiva para quien quiera limitarse a mantener algún o algunos paquetes.

Tras una larga discusión,[14] se produjo el proceso de votación de Resolución General en que la propuesta fue aprobada (Debian.org/vote, 2007). En un mensaje muy posterior se puede encontrar un resumen, con enlaces, a los principales argumentos a favor y en contra (Debian-vote, 2013): las dificultades del NMP y de encontrar un *sponsor*, el hecho de que no todos los que contribuyen quieren ser Desarrolladores, o que era una buena manera de solucionar los problemas del

[14] Puede seguirse gran parte de ella en el hilo que sigue a Debian-newmaint (2006).

Además de los enlaces que se van indicando, véase el «Glosario» de la *wiki* de Debian (Wiki.debian.org, s.f.[an]). Un resumen sobre los diferentes tipos de miembros de Debian puede verse también en Wiki.debian.org (s.f.[ao]).

Application Manager Un miembro de Debian que se asigna a quien inicia el NMP y se asegura de que el proceso se realiza correctamente, reuniendo toda la información necesaria.

Front Desk o New Member Front Desk El equipo encargado de hacer funcionar el NMP y apoyar a los Debian Account Managers (Wiki.debian.org, s.f.[ap]).

Debian Account Manager Son los encargados de crear las nuevas cuentas y mantener la lista de miembros, decidiendo así en última instancia sobre la admisión de los nuevos miembros del Proyecto. Sobre sus funciones puede verse su página en la *wiki* de Debian (Wiki.debian.org, s.f.[n]).

Debian Contributor o Colaborador de Debian Término general que designa a cualquier colaborador de Debian, con una posición formal en el Proyecto o no.

Maintainer o Mantenedor Este término no designa la posición de un miembro en el Proyecto, sino su relación con un paquete. Se refiere a quien (una persona o un equipo) trabaja en los paquetes que Debian distribuye. Esta figura está documentada en Debian.org/doc (s.f.[d]), y se define en un campo en los archivos de control del paquete fuente. Se puede ser mantenedor de un paquete a través de un *sponsor*, siendo Mantenedor de Debian o Desarrollador de Debian (Wiki.debian.org, s.f.[aq]). En este trabajo, las posiciones aparecen completas y en mayúsculas; si nos referimos a quien mantiene un paquete aparece solamente la palabra «mantenedor» en minúscula.

Debian Maintainer o Mantenedor de Debian Posición oficial que supone que se pueden subir algunos paquetes al archivo sin necesidad de *sponsor*. Esta figura existe desde una Resolución General de 2007 (Debian.org/vote, 2007). No son miembros de pleno derecho del Proyecto, en tanto que no pueden votar. En la página *wiki* «Debian Maintainer» (Wiki.debian.org, s.f.[ar]) pueden consultarse sus atribuciones y los pasos necesarios para convertirse en uno. Se recomienda ser Debian Maintainer antes de intentar ser Desarrollador de Debian. Este proceso está actualmente integrado en el NMP.

Cuadro 6.1: Terminología sobre miembros, mantenedores y desarrolladores

Debian Developer o Desarrollador de Debian Equivalente a Miembro de Debian. A partir de la Resolución General «Debian Project Members» de 2010 (Debian.org/vote, 2010) pueden ser *uploading* o *non-uploading*. Son los que tienen todos los derechos como miembros del Proyecto, excepto el de subir paquetes al archivo para los *non-uploading Debian Developers*. El proceso para convertirse en Desarrollador de Debian es el NMP.

Debian Member o Miembro de Debian En la Constitución de Debian, Debian Developer y Project Member son términos intercambiables (Debian-newmaint, 2011a). A lo largo de este trabajo hemos usado sobre todo la expresión Desarrollador de Debian para referirnos a los miembros con todos los derechos, incluyendo a los Desarrolladores que no pueden subir paquetes al archivo, por ser en definitiva la menos ambigua.

New Maintainer o Nuevo Mantenedor Llamado también candidato. Quien está en el *New Maintainer Process* para convertirse en nuevo Desarrollador. Después de la Resolución General de 2010 «Debian project members», el nombre ya no es adecuado, por lo que en 2011 se cambia el nombre a *New Member Process* (véase el mensaje a `debian-devel-announce` «Bits from the New Maintainer^WMember process», (Debian-devel-announce, 2011)). Se evita así también la confusión en cuanto a si se refiere a Mantenedores de Debian o Desarrolladores de Debian. Este cambio no afecta a las siglas, que siguen siendo NMP.

Sponsor o patrocinador Un Desarrollador de Debian que sube un paquete al archivo de Debian cuando su mantenedor no tiene permiso para hacerlo, tras revisarlo (Mentors.debian.net, s.f.[b]).

Mentor Un miembro de Debian que asiste y ayuda a alguien con poca experiencia. No es una posición oficial ni formal. Suelen actuar como *sponsor* o como *advocate*. La página Mentors.debian.net (s.f.[a]) es en realidad una vía de búsqueda de *sponsors*.

Advocate o avalista En el NMP, un Desarrollador de Debian que respalda al New Maintainer o New Member. Suele ser también su *sponsor*.

Uploaders Co-mantenedores de un paquete, que por lo tanto pueden subir nuevas versiones al archivo (Debian.org/doc, s.f.[d]). Se indica en un campo en los archivos de control del paquete fuente.

Cuadro 6.2: Terminología sobre miembros, mantenedores y desarrolladores, parte 2

NMP; en contra, que suponía la creación de miembros de segunda clase, que se complicaba innecesariamente la situación, o que era mejor enfrentar directamente los problemas del NMP. Como seguiremos viendo, problemas que siguieron condicionando el proceso de acceso, y las discusiones y reformas que siguieron.

La clave de esta discusión y de las posteriores está en la cuestión de cómo esta nueva figura afecta en general a la condición de miembro de Debian. En concreto, si facilita o dificulta la incorporación al Proyecto para diferentes tipos de colaboradores, y hasta qué punto el Proyecto Debian es capaz de reconocer apropiadamente esas colaboraciones. Así, encontramos argumentos que insisten en que es un paso adelante en este sentido (Wolf, 2007):

> For long, we have been saying that you don't have to be a programmer to help Debian - But so far, we have been unable to deliver except for this funny French guy. And although the current proposal still focuses on getting a step closer to DDness, and for many people this seems like geared at reducing the frustration when going over the NM process, my impression is that this will help us implement something similar for other areas.

Pero también argumentos de que es una manera de evitar que colaboradores valiosos lleguen a ser miembros de pleno derecho (Debian-vote, 2013):

> The people proposing the GR saw it as widening access. Due to the above two points, for me, it seemed like narrowing it. I could understand reasons for initially putting **technical** restrictions on new contributors, but if we reached the point of fully trusting someone with a package (and therefore root privileges on every machine where it's installed), and giving them a formal status in Debian, I felt that we should already recognise them as members. Though the GR proposers said that it was for people who would not have otherwise have had any status at all, I was worried that the effect was to shut some formally recognised contributors out of membership.

Estas dos citas no representan diferentes objetivos, sino diferentes interpretaciones sobre los efectos de las reformas. En cualquiera de los dos casos, se entiende que no es y no debe ser un cambio aislado, sino que sus efectos han de conectarse con alguna reforma del NMP o de las formas de acceso al Proyecto. El principal riesgo sería entonces que se quedase como una categoría de segunda clase permanente para algunos colaboradores, en lugar de ser un primer paso de la integración en el Proyecto. Así, desde esta reforma a las que analizaremos más adelante parece haberse dado un proceso gradual de separación de los diferentes privilegios, derechos y componentes que formaban parte de la pertenencia a Debian, desde el modelo anterior que la entendía como una alternativa entre tener todos los privilegios o no tener ninguno. Lo planteaba claramente poco después de la Resolución General un Desarrollador de Debian en una propuesta de reforma del NMP: «The suboptimal was considering other kind of contributors, I'm happy with the NM process for selecting DD as it is. But I disliked the DD or nothing approach we had up to now» (Wiki.debian.org, s.f.[as]). En la misma línea, uno de los DAM en aquel momento, el autor de la propuesta que vimos al principio de la sección, señalaba (Jaspert, 2007a), tras mostrar su desacuerdo y votar en contra, que se podría integrar en el NMP existente, de tal manera que se concediese el derecho de subir un paquete en un momento determinado del NMP, y más tarde, al cumplir otros requisitos, se obtuviesen gradualmente otros privilegios y derechos.

Resumiremos ahora el procedimiento y los diferentes pasos en que consistía el NMP en el momento en el que se hizo la propuesta con la que hemos iniciado esta sección:[15]

Solicitud y aval Tras un tiempo colaborando con Debian, y preferiblemente tras al menos 6 meses como Mantenedor de Debian, el candidato realiza una solicitud a través de la página correspondiente (Nm.debian.org, s.f.[c]). Un Desarrollador de Debian ha de avalar al solicitante mediante un correo firmado, en el que describe las aportaciones del candidato y sus planes. El *Front Desk* asigna entonces un *Application Manager* que llevará a cabo el proceso.

[15] Las páginas más relevantes sobre este proceso son Wiki.debian.org (s.f.[at]) y Debian.org (s.f.[y]). La lista de correo relevante es Debian-newmaint (s.f.).

Identificación A partir de una clave criptográfica firmada por al menos uno o dos Desarrolladores de Debian (el número ha variado) en un encuentro presencial cara a cara. Obtener estas firmas supone una comprobación del nombre legal del solicitante (sobre esta compleja cuestión, y otras cuestiones técnicas relacionadas, véase la sección 6.4, *La construcción tecnosocial de la confianza*). Supone conectarse a la *web of trust*, y dicha clave se usa para subir paquetes, mandar algunos mensajes firmados, acceder a los servicios de Debian y votar, entre otras cosas.

Filosofía y procedimientos (Philosophy and Procedures) Como se dice en su página *web* (Debian.org, s.f.[z]): «The Applicant is expected to fit into the Debian community, which is built around the philosophy of Free Software». Los solicitantes han de demostrar que comprenden y aceptan el Contrato Social. Requiere un conocimiento profundo sobre licencias y las *Debian Free Software Guidelines*. En cuanto a los procedimientos y normas estándar (*policies*), el solicitante ha de mostrar que conoce procedimientos básicos como el uso del *Bug Tracking System* o el proceso de publicación de Debian. También ha de conocer y aceptar la «Debian Machine Usage Policy (DMUP)» (normas de uso de máquinas de Debian) (Debian.org, s.f.[p]).

Tareas y habilidades (Tasks and Skills) Los solicitantes han de demostrar que tienen la capacitación técnica para colaborar, sobre todo en lo relativo al empaquetado de *software*. Incluye la evaluación por parte del *Application Manager* de las contribuciones que ya se han hecho a Debian, lo que supone examinar sus paquetes, sus *bugs*, sus parches, etc.

Estos dos últimos pasos pueden y suelen realizarse mediante preguntas que el *Application Manager* plantea al solicitante, y que han de ser respondidas por extenso. Puede consultarse un repositorio de preguntas que se suelen realizar (aunque de ningún modo es obligatorio realizar todas ni limitarse a ellas) en «nm-templates» (Salsa.debian.org, s.f.[c]).

Recomendación El *Application Manager* hace un informe que incluye todo lo realizado por el solicitante, los mensajes de aval, y un comentario del propio

AM, así como su recomendación para aceptar o rechazar al solicitante.[16]

Comprobación del *Front Desk* Comprueba el informe y se aprueba en su caso.

Comprobación de DAM y creación de la cuenta Los *Debian Account Managers* comprueban el informe y deciden si crean la cuenta. Pueden rechazarla o devolverla a *Front Desk* para que sea completada. Si se acepta, se crea la cuenta y se otorgan los permisos y accesos asociados.

El proceso para convertirse en Debian Maintainer es mucho más sencillo y puede consultarse en Wiki.debian.org (s.f.[ar]). Ambos procesos se llevan actualmente a cabo en la página «Debian New Members» (Nm.debian.org, s.f.[a]), que es también la principal página donde comprobar la posición oficial de cualquier miembro o colaborador de Debian.[17]

6.3.1.1 La salida del Proyecto.

Cuando un miembro de Debian quiere abandonar el Proyecto, puede pasar a ser un Desarrollador emérito si sigue una serie de pasos, lo que le permitirá reingresar fácilmente más adelante si decide hacerlo (Debian.org/doc, s.f.[e], cap. 3). Más problemática es la situación que se produce cuando un Desarrollador abandona *de facto* el Proyecto sin retirarse formalmente, sin avisar y sin pedir que otra persona se haga cargo de sus paquetes. Se habla entonces de Desarrolladores MIA (*Missing In Action*). La cuestión es que, dados los diferentes ritmos de desarrollo de la distribución, y el funcionamiento de la doocracia (véase la sección 3.2, *Toma de decisiones: de la meritocracia a la doocracia*), en muchas ocasiones es difícil determinar si alguien está efectivamente MIA, puesto que es, como vimos, cada uno quien decide su trabajo y cuándo realizarlo. Esta situación puede producir paquetes mal mantenidos, de los que, de nuevo a causa de la doocracia, es difícil que otro mantenedor pueda hacerse cargo (véase la sección 3.2.3, *Problemas de*

[16] Los *AM Reports* son enviados a la lista `debian-newmaint`. Los últimos pueden consultarse también en Nm.debian.org (s.f.[a]). Los de los años 2007-2009 se pueden encontrar agrupados en los últimos mensajes de junio de 2009 a la lista `debian-project`, en Debian-project (2009d).

[17] Otras páginas donde consultar datos sobre los miembros y colaboradores son «Debian Member Portfolio Service», en Portfolio.debian.net (s.f.) y Db.debian.org (s.f.[b]). Puede verse también la página *wiki* «Contributors Information Sources», en Wiki.debian.org (s.f.[au]). Y como veremos más adelante, «Debian Contributors», en Contributors.debian.org (s.f.[a]).

la doocracia: ownership, hijacking *y* salvaging *de los paquetes*). Para tratar de solucionar esta situación existe un equipo en Debian llamado precisamente MIA (Wiki.debian.org, s.f.[av]) cuya función es detectar estas situaciones y reflejarlas en la base de datos de MIA. Tras un proceso de varios pasos, el equipo puede dejar huérfanos los paquetes afectados para que se haga cargo de ellos otro Desarrollador, y avisar a los DAMs, una de cuyas funciones es suspender la cuenta de Debian del Desarrollador implicado. Son éstos procesos que se ha intentado ir clarificando y automatizando progresivamente,[18] de tal modo que se evite un trabajo manual, difícil y controvertido en el equipo MIA, sobre lo que hay un acuerdo generalizado aunque se discutan los detalles.[19]

La importancia concedida al hecho de limitar el número de Desarrolladores de Debian inactivos con una cuenta funcional remite a tres elementos. En primer lugar, las tensiones propias de la doocracia, que otorga un gran poder al mantenedor oficial de un paquete y hace difícil la retirada de derechos respecto al mismo, al tiempo que hace en muchos casos difícil determinar si un Desarrollador está efectivamente inactivo (por ejemplo, si no actualiza sus paquetes ni resuelve los *bugs* pero expresa que a su juicio *esa* es la manera correcta de proceder); en segundo lugar, el problema de confianza y seguridad que supone dada la capacidad de un Desarrollador para subir un paquete al archivo de Debian (paquete que puede por lo tanto ser instalado en cualquier ordenador que tenga instalado Debian) o acceder a sus máquinas, cuando esa confianza no está relacionada con una interacción regular con el Proyecto y la comunidad; y por último, la estrecha asociación que expresa entre miembros del Proyecto y los componentes técnicos de la distribución, los paquetes, lo que a su vez nos remite a la caracterización de Debian como un colectivo sociotécnico, como veremos en la sección 6.3.5, *Un colectivo de humanos y no humanos*. Y es que la labor de MIA consiste en detectar Desarrolladores desaparecidos, pero sobre todo paquetes no mantenidos. Su función es facilitar que nuevos mantenedores puedan hacerse cargos de esos paquetes, como co-mantenedores o tras ser declarados esos paquetes «huérfanos» y puestos en adopción. Lo que importa en última instancia es la calidad del paquete, pero ésta depende y es inseparable de quién lo mantenga. Un Desarrollador no activo suele

[18] Pueden verse por ejemplo Debian-devel-announce (2005, 2007) y Debian-project (2009e), o Jaspert (2007b).

[19] Sobre esto véanse Debian-project (2009e,g,f), aunque sigue siendo necesaria la intervención manual de los miembros de MIA y DAM.

significar paquetes mal mantenidos. La integración de un paquete en Debian depende de su relación con el mantenedor, del mismo modo que, en la mayor parte de los casos, la integración de un nuevo miembro depende de su relación con uno o varios paquetes.

6.3.1.2 Mentorización, confianza, y doocracia.

Querría detenerme ahora en dos breves consideraciones, muy relacionadas entre sí, sobre el sentido de este proceso. En primer lugar, si bien el NMP constituye para los Desarrolladores uno de los momentos más importantes en su relación con el Proyecto y la comunidad Debian, pudiendo incluso considerarse como rito de paso (Coleman y Hill, 2005, pág. 280), es necesario enfatizar que este proceso no es un momento inicial ni final en esa relación, sino que se produce siempre a medio camino. Así, se debe solicitar la entrada cuando ya se tiene una considerable experiencia previa en la colaboración con Debian: «The ideal New Maintainer applicant is one who the Application Manager would have assumed was already a Debian Developer: already active and doing useful work in Debian» (Wallach *et al.* 2005, pág. 9). De igual modo, un Desarrollador sólo debe avalar a un solicitante del que piense que ya está listo para ser Desarrollador, como se establece en la página de Debian «Advocating a prospective member»: «You should *only* advocate someone if you think that they are ready to be a project member. This means that you should have known them for some time and can judge their work. It is important that prospective members have been working in the project for some time» (Debian.org, s.f.[aa]). O como aparece en un correo a `debian-devel-announce` de los responsables del proceso: «When you advocate a person, you are saying that they need and should get unsupervised upload rights on the entire archive, right now» (Debian-devel-announce, 2009).[20]

Encontramos así una concepción del NMP como momento puntual (en la medida en que puede serlo) cuya función principal no es el aprendizaje sino la obtención de una cuenta. Un proceso que sólo debe emprender quien ya esté listo y que se ha de centrar en lo que se ha hecho previamente. Una consecuencia de ello es que se debe también rechazar pronto a quien inicie el proceso sin estar lo suficientemente preparado, a fin de evitar perder el tiempo.[21] Este rechazo o

[20] Véase también Debian-devel-announce (2010c).
[21] Véanse Debian-devel-announce (2009) y Debian-devel-announce (2010c). También la *lightning*

cancelación de la solicitud no implica nada sobre la continuidad de la colaboración con Debian.

Pero esta concepción coexiste con otra que entiende el NMP también como proceso de formación y aprendizaje, si bien más en relación a lo que tiene que ver con el paso «Filosofía y Procedimientos» que con el más técnico relativo a «Tareas y Habilidades». Así por ejemplo, para Coleman el NMP supone «a centripetal force of ethical enculturation» (Coleman, 2013, pág. 141).[22] O como dice un Desarrollador de Debian:

> **mones:** Ha habido una especie de renovación generacional. La mayoría de la gente ha pasado por el NMP, que no pasaba antes, el núcleo duro no había pasado. Pasar por NMP te da una especie de adherencia a hacer las cosas de manera ordenada y en colaboración, no porque sea mi visión y lo voy a hacer así. Eso en un proyecto pequeño vale, pero mil personas no hablan de todo todos los días. Es imposible materialmente. [...] El hecho de pasar por el NMP te da una idea de cómo hacer las cosas, no sé si más oficial, más abierta quizás.

Una colaboradora que inició el NMP poco después de la DebConf en la que se realizó la entrevista, y se convirtió más adelante en Desarrolladora, insistía en esta consideración del NMP como proceso de aprendizaje:

> **monica:** La gente te pregunta, ¿y en qué estás interesado? Y lo que me pasaba a mí, y por lo que han dicho le pasa a mucha gente, es que tú quieres ayudar, pero no estás interesado en nada en concreto, sino que quieres ayudar y punto. Y no encuentras tu lugar, porque Debian no funciona así. [...]
>
> Cuando haces este proceso [el NMP], no es que se suponga que lo sabes todo, sino que vas a aprender mientras haces el proceso. [...] Yo eso

talk «Bits from the AMs, FDs, DAMs, NMs» (Debconf.org, s.f.[c]) que resume los resultados del «NM AM FD DAM BOF» en Debconf 9 (Debian-newmaint, 2009).

[22] Para una perspectiva más amplia sobre esta cuestión, no centrada exclusivamente en el NMP, véase Coleman y Hill (2005).

> no lo sabía, yo pensaba que era un examen, no un proceso en el cual también estás aprendiendo.

Es normal que asistir a una DebConf sea el acontecimiento que influye decisivamente en que un colaborador decida convertirse en Mantenedor o Desarrollador de Debian.[23] Esto muestra la importancia de los encuentros presenciales para la vida de la comunidad Debian, como vimos en la sección 6.2.1, *DebConfs*. Como vimos en el capítulo anterior, la voluntad abstracta de colaborar y devolver algo a la comunidad es en numerosas ocasiones previa al interés concreto por alguna forma específica de colaboración. El NMP entendido como proceso de aprendizaje, o mejor como un elemento más en ese proceso más amplio, puede servir entonces como forma de organizar la obligación de recibir y como forma de integración de nuevos colaboradores.

Por su parte, un Desarrollador de Debian que en el momento de la entrevista era *Application Manager* expresaba así su opinión de que el NMP debería ser un proceso con un fuerte componente de mentorización:

> **dato:** Ahora tenemos un proceso un poco muy rígido. Tienen que apuntarse y se les asigna un tutor, más que tutor un mentor, o más que mentor un examinador, sí, que les hace un montón de preguntas, y las tiene que responder, y es como un examen que dura varios meses, y es muy aburrido. [...] Yo creo que debería estar basado en que algunos Desarrolladores hayan trabajado muy cercanamente con una persona. [...] Nuestro proceso debería estar orientado, más que a la esponsorización, a la mentorización, en el que un Desarrollador se hace cargo personalmente del proceso de aprendizaje de un estudiante, o como lo queramos llamar. Creo que es el modelo que funcionaría mejor. [...] Es una de las tareas más importantes que uno pueda realizar en Debian, porque al fin y al cabo es lo que va a determinar cómo van a trabajar los nuevos Desarrolladores, qué prioridades van a tener.

[23] Véase por ejemplo Debian-newmaint (2012).

Sería entonces un proceso en que se pueden dar diferentes formas de mentorización. Propongo entender que la primera concepción, el NMP como examen y comprobación de la competencia, tiene de hecho una relación muy estrecha con la doocracia, en tanto que los privilegios y el acceso siguen al trabajo ya realizado, son una consecuencia del mismo. El aval y la creación de la cuenta serían el reconocimiento de una situación alcanzada por el nuevo miembro a partir de sus contribuciones efectivas. Por otra parte, la concepción del NMP como lugar de aprendizaje tiene más que ver con la meritocracia, en tanto se entiende que el solicitante adquiere y demuestra en ese proceso el merecimiento que le permitirá tomar decisiones más adelante, una vez haya obtenido el estatus de Desarrollador. Del mismo modo que doocracia y meritocracia no son procesos excluyentes, tampoco lo son en modo alguno las dos concepciones del NMP discutidas.

La segunda cuestión se refiere también a una de las funciones del NMP: la generación de confianza en los futuros Desarrolladores, o la confirmación de que son dignos de confianza. Una Desarrolladora y candidata a DPL lo indicaba con mucha claridad en un correo electrónico en la lista `debian-vote`: «We need to be able to **trust** that person with their responsibilities, the whole point of the NM process is to allow us to trust» (Debian-vote, 2010a). Otro de los candidatos señalaba también el lugar central de la confianza: «The reason to be somehow strict before accepting someone as a DD is essentially trust. A DD will be able to upload any package to our distribution, so we should trust his/her judgement in when to (not) use such a privilege» (Debian-vote, 2010b).

La cuestión de la confianza remite a una tensión interna que se produce en el proceso del NMP. Por un lado éste ha de suponer una barrera, un filtro, frente a potenciales Desarrolladores que no están lo suficientemente capacitados. Esto invita a hacer difícil el proceso, pero al mismo tiempo a entenderlo no como proceso de formación sino como una comprobación de lo ya realizado. Por otro lado, el NMP es también parte del proceso más amplio de reclutamiento, de integración activa de nuevos colaboradores y Desarrolladores. Esto invita a que sea también usado como elemento de formación y mentorización. La tensión proviene entonces de dos problemas a evitar: que personas que no estén preparadas obtengan derechos y privilegios, y que no los obtengan las que sí lo están. En definitiva, de lo que se trata es de construir la confianza por parte del Proyecto en los nuevos

Desarrolladores. El NMP es buena parte de la respuesta a la pregunta sobre quién es miembro de Debian no sólo porque explicite quién es miembro formal de Debian (lo que no deja de ser una tautología), sino porque define qué se espera de un miembro de Debian.

El reconocimiento de esta tensión nos ayuda también a deshacer una paradoja en relación al NMP. En muchas de las entrevistas realizadas, así como en muchos relatos sobre el NMP, una observación frecuente ha sido el tiempo y esfuerzo que es necesario invertir en el proceso, lo que aparentemente contradice la apertura del Proyecto de la que venimos hablando, y que también aparece en esas entrevistas y relatos. Hemos expuesto ya lo suficiente para ver que la apertura se refiere a la visibilidad de los procesos y a la recepción de las colaboraciones más que a la facilidad para alcanzar la posición de Desarrollador. Pero como veremos en las próximas secciones, esa apertura, la necesidad de reconocer propiamente todas las colaboraciones y aportaciones, en tensión con la necesidad de construir la confianza de la que acabamos de hablar, ha sido una de las motivaciones fundamentales en los cambios y la dinámica del NMP.

6.3.2 Reacciones al anuncio sobre «Developer Status».

Volviendo ahora al anuncio de 2008 sobre «Developer Status» citado anteriormente, se produjeron a partir del mismo en el Proyecto una serie de discusiones y propuestas alternativas que se pueden seguir sobre todo a partir de los siguientes hilos en la lista `debian-project`: el que empieza en el mensaje Debian-project (2008a), donde se contesta directamente al anuncio que se publicó en la lista `debian-devel-announce` (a la que no se puede responder; las discusiones relativas a anuncios producidos ahí suele realizarse en `debian-project` o `debian-devel`), y el que empieza en Debian-project (2008b), donde se hace una propuesta de reforma alternativa que se seguirá discutiendo en los meses siguientes. Como ejemplo de algunas respuestas (y de nuevo alternativas) en los *blogs* de destacados desarrolladores, pueden verse por ejemplo Hess (2008), Nussbaum (2008b) y Hertzog (2008). Una respuesta del autor de la propuesta original, en la que contesta algunas críticas y aclara algunos elementos, puede leerse en Jaspert (2008).

Resumiendo, las críticas más generalizadas que suscitó este anuncio tenían

que ver tanto con la forma como con el contenido del mismo, incluso más con la forma que con el contenido. Respecto a lo primero, se critica la forma de tomar la decisión y de anunciarla, como una decisión ya tomada por una sola persona (o un pequeño grupo), en lugar de buscar el consenso a través de una discusión general previa, de tal modo que todo el proyecto pudiera participar en la misma. Como ejemplo, puede verse el siguiente correo de uno de los hilos citados (Debian-project, 2008c):

```
To: «Debian Project List» <debian-project@lists.debian.org>
Subject: Re: Developer Status
From: «Margarita Manterola» <margamanterola@gmail.com>
Date: Wed, 22 Oct 2008 22:23:24 -0200

On Wed, Oct 22, 2008 at 7:33 PM, Joerg Jaspert <joerg@ganneff.de> wrote:
> We plan to integrate DM more closely into the NM process/system
> while keeping the spirit of easing entry into Debian for newcomers.
> At the same time we add a separate track for less-technical
> contributors.

I have been wanting this for a long time, I totally agree with what's in
the mail, and I'm really happy that this has been thought about and written
about, however, I'm quite disappointed on this being informed as a done
thing, without the project as a whole being asked for an opinion.

It would have been better if this had been presented as a DEP, a GR, or
something like that. :-\

I know that DAM's and Front Desk and the like and in charge of accounts,
but creating 2 new categories for people to participate in is quite a change,
and I think the whole project should be involved in such a decision, as
we were involved in the decision regarding Debian Maintainers.
```

Se produce así una tensión entre la doocracia por un lado, y el consenso y la democracia por otro como formas de decisión, a la que se une, como hemos visto, la inquietud provocada por la acumulación de posiciones en el proyecto y la concentración de poder que supone (sobre todas estas cuestiones, véase la sección 3.2.4, *Concentración de poder y los límites internos de la doocracia*). El autor del anuncio que estamos discutiendo, el Desarrollador de Debian Joerg Jaspert (gan-

neff),[24] es *Debian Account Manager* o DAM (véase el cuadro 6.1, *Terminología sobre miembros, mantenedores y desarrolladores*), y por ello está facultado de hecho para decidir qué cuentas se crean en Debian, y por tanto quién es Miembro de Debian.[25] Significativamente, este Desarrollador es también miembro del equipo FTP Master,[26] con capacidad para decidir en última instancia qué paquetes se incluyen en la distribución Debian. Como veremos en la sección 6.3.5, *Un colectivo de humanos y no humanos*, la integración de nuevos miembros y nuevos paquetes son dos de las cuestiones que suelen generar controversia en Debian, y están relacionados entre sí. Si la doocracia faculta a determinados Desarrolladores para decidir sobre determinadas áreas, también se considera que algunas de sus implicaciones deberían ser más ampliamente discutidas.

En cuanto al contenido, aunque muchos de los participantes están de acuerdo en el fondo de la cuestión, las principales preocupaciones se refieren al aumento de la burocracia y la complicación del proceso, así como a algunos detalles de la implementación propuesta, de los privilegios asociados y de la terminología usada. Puesto que al final la propuesta no se implementó, no es necesario entrar en esos detalles. Pero sobre todo, en la introducción de dos clases de miembros con estatus y privilegios diferenciados. Como vimos, la propuesta de ganneff incluía explícitamente dos clases de Desarrolladores de Debian, una con derecho a subir paquetes al archivo, y la otra sin ese derecho. Explícitamente negaba también que eso supusiese una distinción entre un nivel de primera clase y otro de segunda clase. No todo el mundo estaba de acuerdo en esto último, mientras que otros señalan que esa distinción, de una manera u otra, ya existía, que la propuesta permite en todo caso participar más:

> **vorlon**: One of the complaints a number of people have raised about that, is that it creates separate classes of developers and there is less equality. [...] And what we are doing instead is broadening. We already

[24] Puede leerse una entrevista con él en Hertzog (2012c).

[25] Pero no, sin embargo, quién obtiene una cuenta en las máquinas de Debian. Éstas son responsabilidad del equipo DSA, *Debian System Administrators* (Dsa.debian.org, s.f.), y suya es la decisión sobre las cuentas en las mismas. Sobre esto, puede verse el correo a la lista `debian-project` en Debian-project (2009h). Esto muestra claramente la granularidad del funcionamiento de la doocracia.

[26] Sobre el papel de este equipo puede leerse Brockmeier (2010).

> have two classes. Before we had the Debian Maintainer idea, we had people who are Developers and people who are not Developers. So what we are doing is giving some of the people who are not Developers a way to participate, to give them the opportunity to contribute fully, without as many obstacles.

Y es que se puede entender como una manera de ir separando los diferentes derechos y privilegios, de tal modo que se vayan obteniendo progresivamente, de manera diferenciada:

> **dondelelcaro:** The basic idea, as far as I'm concerned, is to try to get people who want to contribute, able to contribute as fast as possible. And part of the problem with the NMP, as it is currently, is that it is a process that is designed to give you the keys of the universe, basically. When you [are a Debian Developer] you can do anything, basically, in Debian, that you want. You can obtain root on any machine that runs Debian, so this is a serious thing and obviously it is important to be careful about granting that.
>
> Everything else isn't as potentially world altering. So I think we do need to start adding levels. The problem with it is that there is a perception that somebody who is on one of these levels is somehow a second class contributor. I don't think this a fair perception but it is something that happens.

Se trataría, entonces, de ir construyendo la confianza, que como hemos visto es un componente fundamental del NMP, poco a poco:

> **karora:** I think it's a good idea. It enables the development of a trust relationship in small steps, which I think is entirely appropriate. The Debian Developers are in a position of trust, because of the nature of the way software is installed on Debian systems. [...] You need to know that people who you allow to do that are trustworthy.

Más allá de las críticas o de las propuestas alternativas, las discusiones en torno a toda esta cuestión apuntan a problemas considerados como tales por muchos miembros de Debian. Lo que es común es la necesidad de repensar de alguna manera el NMP. La preocupación básica, más allá de problemas como la duración o complejidad del mismo (Nussbaum, 2009), es la necesidad compartida de reconocer el trabajo de los colaboradores no técnicos, en tanto que el NMP estaba diseñado en su origen pensando en alguien que colabora empaquetando *software*. Se reconoce que hay colaboradores que deberían poder votar, incluso aunque no puedan subir paquetes al archivo de Debian. Para ello, de alguna manera, deberían poderse separar estos derechos, organizar los derechos y privilegios (sobre todo el de *upload*) de manera más granular, aunque sobre esto no hay realmente consenso. También existe, paralelamente, una preocupación por la calidad técnica de las contribuciones, y quejas sobre la calidad técnica del trabajo de algunos Desarrolladores y Mantenedores. En relación con esto, también encontramos la preocupación respecto al papel de los miembros inactivos, que siguen teniendo derecho a voto y a subir paquetes. Aunque no entramos en ello, existe una dificultad añadida en cuanto a la dificultad técnica de implementar técnicamente los derechos y privilegios, y de integrarlos con los niveles de permisos y acceso a la infraestructura y los servicios de Debian, que se basan muchas veces en un modelo de todo o nada.

En el fondo, la cuestión que yace bajo toda esta controversia es la de quién es y quién debería ser miembro de Debian. Como veremos, esta pregunta es en realidad doble: si se discute sobre el procedimiento formal para entrar en el Proyecto Debian, es sobre el trasfondo de su conexión con la cuestión de quién es miembro de la comunidad Debian y de qué supone esta pertenencia. En cualquier caso, en todas estas discusiones se ve con claridad que lo que supone la pertenencia al Proyecto son fundamentalmente los derechos de subida de paquetes al archivo y de voto, y que ambos están en tensión con la necesidad de reconocimiento de las diversas formas de contribución: por un lado, todos los que contribuyen significativamente deberían ser miembros del Proyecto (y poder participar de su gobierno a través del voto); por otro, sólo aquellos individuos técnicamente cualificados deberían tener la posibilidad de subir paquetes al archivo o modificarlos. Pero al mismo tiempo, muchos piensan que ser miembro de Debian significa precisamente poder hacer esto último (puesto que lo que crea Debian no es más que una

distribución), y que introducir limitaciones supondría necesariamente distinguir miembros de primera y de segunda clase. Significativamente, estos dos derechos (voto y subida de paquetes) están estrechamente relacionados con dos de las formas de gobierno centrales en Debian (véase el capítulo 3, *Formas de gobierno. Doocracia, agencia y apertura*): si el derecho de voto tiene que ver con la democracia, el de subir paquetes (junto con otros como el acceso a las máquinas y servicios del Proyecto) tiene que ver con la doocracia, con la capacidad efectiva de actuar en el Proyecto. De ahí posiblemente la reticencia de algunos a la hora de aceptar que pueda haber miembros sin la capacidad de producir paquetes para la distribución. Hay que señalar también que, como hemos visto en la sección 6.3.1, *El* New Maintainer Process, ambos derechos requieren la confianza por parte del Proyecto.

La discusión sobre esta propuesta concreta desembocó en una Resolución General sobre la misma, que se acabó de votar el 14 de diciembre de 2008 (Debian.org/vote, 2008b), propuesta en primer lugar el 24 de octubre de 2008 por un Desarrollador de Debian (Debian-vote, 2008a), y que propone que se suspendan las decisiones anunciadas por ganneff. En la larga discusión que sigue, la cuestión principal no es, en consonancia con lo que hemos visto, la conveniencia o no de los cambios propuestos, sino el proceso de toma de decisión y la forma de anunciar las medidas. Se llega a cuestionar y discutir si DAM puede realizar tales cambios que afectan a un proceso básico en Debian, sin contar con un amplio consenso o una Resolución General.[27]

Tras alguna enmienda y propuestas de alternativas, así como comentarios sobre el significado de cada propuesta, se realiza la votación sobre las siguientes propuestas:

> **Choice 1:** Ask the DAMs to postpone the changes until vote or consensus.
>
> **Choice 2:** Invite the DAM to further discuss until vote or consensus, leading to a new proposal.

[27] Sobre esto, véase el sub-hilo «DAM has no competency to make changes to membership structure (was: Call for seconds: Suspension of the changes of the Project's membership procedures.)», a partir de Debian-vote (2008b).

Choice 3: Ask the DAMs to implement the changes.

Choice 4: Further Discussion.

En línea con las reacciones que hemos ido viendo, la opción ganadora fue la segunda, «Invite the DAM to further discuss until vote or consensus, leading to a new proposal»,[28] de tal modo que los cambios no se implementaron, esperando otro proceso con más consenso. Pero reconociendo al mismo tiempo que había problemas con el actual proceso de incorporación a Debian como miembro de pleno derecho.

Incluso después de esta votación, todas estas cuestiones siguieron interesando al proyecto, como se puede ver por ejemplo en el hilo que sigue a Debian-project (2009i) (una nueva propuesta de reforma por parte de un DD), o a Debian-vote (2009), una pregunta a los candidatos a DPL en la elección de 2009 sobre el procedimiento para ser miembro de Debian y las propuestas de reforma citadas. De hecho, preguntas sobre estas cuestiones han estado presentes en todas las campañas para DPL que he seguido.

6.3.3 Diversidad en Debian: otros procesos de reforma.

Como vimos en la sección 1.2, *Origen de la investigación y acceso al campo*, en las páginas *web* de Debian se deja claro desde el primer momento que colaborar con el Proyecto no requiere tener una posición oficial (Mantenedor de Debian o Desarrollador de Debian), ni se limita al trabajo de empaquetado (Debian.org, s.f.[d],[f]). En la práctica hay muchas formas diversas de colaboración y muchos niveles diferentes de participación en la vida social de Debian, independientemente de la posición oficialmente reconocida que se tenga. Ésta no es un buen indicador de la participación en la vida del Proyecto, ni del nivel de colaboración con el mismo. Como vimos en la introducción, y de acuerdo con la doocracia y la apertura que caracterizan a Debian, empezar a colaborar con el Proyecto no tiene más condiciones que empezar a hacerlo.

Las formas de contribución en Debian van pues más allá del empaquetado de *software*. Sin embargo, normalmente se ha entrado en el colectivo asociándo-

[28] Un análisis interesante de los resultados se puede encontrar en Debian-vote (2008c).

se a una pieza de *software*, dado el perfil eminentemente técnico del NMP y la importancia del paso relativo a Tareas y Habilidades, donde había que demostrar que se poseían las capacidades y conocimientos técnicos necesarios, y que se conocen las sutilezas técnicas de la distribución. Aparece como fundamental no sólo contribuir al funcionamiento del Proyecto, sino ser capaz de establecer esa relación concreta con esos objetos técnicos que son los paquetes de *software* (véase la sección 4.5, *El empaquetado como práctica sociotécnica*.) Pero el caso es que hay gente que contribuye, algunos de manera muy importante y esencial para el Proyecto, en otras actividades y necesidades, y no es capaz o no está interesada en construir ese tipo de relación, que puede llegar a ser muy complicada. De ahí la necesidad de reconocer otros tipos de miembro de Debian.

Por eso ha sido una discusión recurrente en Debian la necesidad de reformar el proceso de pertenencia formal de tal modo que pudiesen ser reconocidos como miembros de pleno derecho quienes colaboran regularmente en actividades con un perfil no tan técnico (se suelen mencionar entre otras la traducción, diseño, comunicación, o la asesoría legal). Es cierto que existen muchas tareas técnicas que no suponen directamente mantener paquetes, como arreglar *bugs*, labores de *Quality Assurance*, o mantenimiento de la infraestructura, por ejemplo, pero la mayor parte requiere el conocimiento de cómo son y se preparan los paquetes. Veamos algunas perspectivas de los Desarrolladores entrevistados al respecto:

> **mvz:** People like translators, also documentation writers, artists, and all kinds of people who are actively involved in the Project and care for the Project, and do work on and for the Project. I wish we could make those full members in the sense that they can vote, and even if they may not have the technical background or necessity to actually upload packages. [...] Technical privilege [...] you need to earn it, you need to prove that you can do it, [...]. Ok, we, I would want more people to be part of the project proper than just, like most of us existing members are of the technical kind [...]. I would want to make it easier to become a full member rather than say, you are half a member or something. I don't like the idea of being like a hierarchy of members. I don't like that.

Se trata entonces de reconocer como miembros completos, no de segunda clase, a los que contribuyen independientemente del carácter técnico o no de esa contribución. Si los miembros de Debian se caracterizan casi exclusivamente por su perfil técnico, es en parte debido a las características del NMP, que hacen muy difícil la incorporación de colaboradores con otro perfil:

> **agi:** [Sobre el NMP] El proceso es muy largo, es muy tedioso, y por el camino se queda gente muy valiosa que se harta. Y luego el problema de ese proceso es que es un proceso pensado para mantenedores, para técnicos, no para otra gente que pueda aportar mucho a Debian, que hoy lo hace de forma no oficial, no como Desarrollador Debian, sino que es un voluntario que aporta, y entonces el proceso no es válido para gente que traduce, que crea gráficos, fondos de pantalla, iconos, sonidos, ideas de marketing o de promoción, o ... Todos esos perfiles no están contemplados en el proceso de incorporación a Debian, [...] habría que reformarlo un poco, en dos aspectos. En el de extensión en el tiempo y dificultad, un poco, y en el de enfoque. No todo el mundo que puede aportar a Debian es un Desarrollador, o un mantenedor de paquetes.

Pero hay un acuerdo generalizado respecto a la necesidad de reconocer propia y formalmente esas aportaciones:

> **dato:** Sí, el problema es, creo que todos estamos de acuerdo en que a las personas que no contribuyen con paquetes hay que reconocerlas. Bueno, a ver, casi todos estamos de acuerdo en que hay que hacer algo respecto a las personas que contribuyen traducciones sólo o documentación, o iconos o arte. Hay algo de división de opiniones sobre si tienen que ser miembros de pleno derecho o no, porque hay gente que dice que no sería bueno que tuviera derecho a voto una persona que a lo mejor no sabe de los tecnicismos de Debian, o que no se ha comprobado donde están situados respecto a la filosofía del *software* libre ...

En el temor a que puedan votar en Resolución Generales personas que no tengan un perfil técnico encontramos una paradoja. Porque habíamos visto que para los Desarrolladores de Debian, si hay algo que no debe ser sometido a votación son precisamente las decisiones técnicas (véase la sección 2.2.1, *Concepciones* emic *de lo técnico y lo social*). Como dice un Desarrollador:

> **dondelelcaro:** The main reason why voting is such an important thing is because it combines the social aspects and also the technical aspects all in one. Because it is extremely difficult to, the secretary basically, when a vote happens, shouldn't be required to separate out the technical aspects from the social aspects. So, a common fear of people is that we will have a General Resolution overriding or making a technical decision, and there will be people that have the ability to vote who do not have the technical background necessary to vote. Now, that's never happened yet in the project.

Y es que la necesidad de adquirir los conocimientos técnicos para tomar una decisión es algo también común a quienes tienen un perfil técnico, debido a la especialización necesaria. Eso ocurrió por ejemplo en la votación que analizamos sobre la inclusión de *firmware* no libre (véase la sección 2.2.4, *Un ejemplo: el* Firmware *y la inscripción de programas de acción*):

> **gregoa:** Voting is the key for membership. So without voting you are not really a member, and if you are a member you must have a right to vote. [...]
>
> [Sobre la Resolución General sobre el *firmware*] I think it is a perfect example, because, sure you need a basic understanding, but you wouldn't do something with Debian anyway if you don't have this. You don't need to know the technical details, how to get this firmware out, in order to have your opinion about it. And I mean, I was allowed to vote about it, and I don't do anything about kernel stuff, so why should the translators?

> **marga:** Mirá, el tema del *firmware* no libre es un tema que es muy complejo. Yo como Desarrolladora de Debian tampoco tenía mucha idea. Tuve que leerme todos los hilos que había en `debian-vote` en ese momento para tener una idea de lo que estaban hablando, porque es algo bastante específico al *kernel*. Me parece que de la misma manera alguien que no fuera responsable de paquetes podría leerlo y tomar una decisión informada.

El reconocimiento de todas las aportaciones no es sólo importante para los afectados, también lo es para la salud y riqueza del Proyecto y la comunidad:

> **amaya:** Si tú tienes mucho interés en votar tendrás que pasar un proceso que es un poco largo y un poco pesado para hacerte Desarrollador, [...] no sólo hace falta gente técnica en un proyecto, hace falta gente con otras destrezas. [...] Cada vez está más valorado que la gente que entra en Debian no solamente sean técnicos *hardcore*, queremos gente que cada vez sea más polivalente.

Como vimos en la sección 6.1.1, *Los sentidos de la comunidad*, se podrían distinguir hasta cuatro niveles de la comunidad Debian atendiendo a su extensión. Lo que descubrimos en estos niveles son las tensiones entre los diferentes sentidos del término «comunidad» que funcionan en Debian. No debemos considerar a ésta como un grupo dado y bien definido, sino como una realidad que está construyéndose simbólicamente. Por tanto, no se debe entender esta división como una división entre diferentes categorías de la misma comunidad, sino como el reflejo de diferentes posibles sentidos del término. Estos sentidos serían progresivamente más inclusivos, y sus límites más imprecisos.

En primer lugar, estarían los que tienen una posición oficial en Debian, los Desarrolladores de Debian y en menor medida los Mantenedores de Debian (como vimos, en algunos contextos e interpretaciones no todos ellos, sino los están presentes en los canales de interacción y mantienen relaciones interpersonales). En segundo lugar, los colaboradores, aquéllos que sin ser miembros formales contribuyen regularmente y son conocidos en el Proyecto. También se entiende en

determinados contextos que los usuarios de la distribución forman parte de la comunidad. Además, podríamos añadir un cuarto grupo, la «comunidad del *software* libre». Este sentido sería quizás el más vago, pero a fin de cuentas es en este campo donde se define el Proyecto, y este sentido queda recogido en el Contrato Social.

La tensión relevante en el contexto del análisis del NMP y sus reformas se sitúa entre los dos primeros sentidos distinguidos, el de los miembros oficiales con un fuerte perfil técnico (entendiendo técnico como lo relativo al empaquetado de *software*) y el del resto de colaboradores con un perfil distinto (y que por eso mismo tenían muy difícil llegar a superar el NMP). Así que en octubre de 2010 se produjo una votación para permitir la pertenencia formal al Proyecto de colaboradores sin ese perfil técnico, de tal modo que pudieran llegar a ser Desarrolladores de Debian y ser incluidos plenamente en la comunidad, sin distinciones.

6.3.3.1 Dos Resoluciones Generales: «Debian project members» y la Declaración de Diversidad.

El texto que se vota en esa Resolución General es el siguiente (Debian.org/vote, 2010):

> **Choice 1: welcome non-packaging contributors** The Debian project aims at producing the best free operating system. To that end, the project benefits from various types of contributions, including, but not limited to: package maintenance, translations, infrastructure and website maintenance, porting, bug triaging and fixing, management activities, communication, testing, legal advice, quality assurance, etc.
>
> The Debian project acknowledges that:
>
> - To pursue Debian goals, package maintenance as well as a wide range of other technical and non-technical contributions are all valuable.
> - Active contributors of non-packaging work, who share Debian values and are ready to uphold Debian Foundation Documents, deserve the opportunity to become Debian Developers.

> The Debian project, therefore, invites the Debian Account Managers to:
>
> - Endorse the idea that contributors of non-packaging work might become Debian Developers, albeit without upload access to the Debian archive.
> - Establish procedures to evaluate and accept contributors of non-packaging work as Debian Developers.
> - Initiate the appropriate technical measures to enable contributors of non-packaging work, who get accepted as Debian Developers, to participate in Debian decision making and to access Debian infrastructure.

Esta Resolución fue aprobada por una amplísima mayoría. Como se ve, se trata sobre todo de explicitar el reconocimiento a todo tipo de colaboraciones y la posibilidad de convertirse en Desarrollador sin dedicarse a empaquetar *software*. Esto permitirá a colaboradores importantes dos derechos fundamentales de los Desarrolladores: participar en las votaciones y las decisiones importantes, y acceder a la infraestructura y los servicios de Debian. Pero las medidas concretas para implementar esta figura se dejan en manos de DAM. En realidad, se puede decir que esta Resolución General no era estrictamente necesaria, en tanto que ya existían algunos Desarrolladores sin perfil técnico y que no mantienen paquetes, y ésta es una posibilidad que está recogida en la misma Constitución de Debian. Además, a fin de cuentas, es decisión del *Application Manager* cómo se lleva a cabo el NMP, y de los DAMs la aceptación final y la creación de la cuenta.[29] Bastaría confiar en que no subiesen paquetes al archivo que pudiesen ser perjudiciales, pero esto es algo que con mayor o menor extensión se espera de todos los Desarrolladores. Sin embargo, la votación de 2008 parando la decisión de DAM vista anteriormente hacía recomendable esta votación para solicitar a DAM por acuerdo general un procedimiento formalmente reconocido, mostrando claramente que hay un consenso al respecto. Esta votación permite, además de explicitar ese reco-

[29] Sobre esto puede verse el correo con el que el proponente de la votación y en en ese momento DPL iniciaba el proceso de Resolución General: «GR: welcome non-packaging contributors as Debian project members», en Debian-vote (2010c).

nocimiento, hacer más visible esa posibilidad y animar a los posibles solicitantes con otros perfiles a iniciar el proceso.

Una de las cuestiones interesantes en la discusión relativa a la votación[30] fue la relativa al nombre que se daría a estos nuevos Desarrolladores. La propuesta inicial era la creación de una nueva denominación, *Debian Contributors* o Colaboradores de Debian, pero la propuesta final aprobada mantenía el nombre de Desarrollador de Debian para esta nueva figura. Relacionada con esta preocupación, algunos manifiestan estar de acuerdo respecto a la aceptación de Desarrolladores que no mantengan paquetes, pero también estar preocupados por la posible creación de miembros de segunda clase (Debian-vote, 2010e,f). Ésta es la razón principal para no establecer un nombre distinto: todos los miembros de Debian, que tienen derecho a votar en las Resoluciones Generales del Proyecto, son denominados Desarrolladores de Debian, especificándose sólo cuando sea necesario si tienen o no la posibilidad de subir paquetes al archivo.

En un correo a `debian-devel-announce` de noviembre de 2010 aparece la primera implementación concreta de este procedimiento (Debian-devel-announce, 2010d). El único cambio respecto al NMP tradicional consiste en el paso relativo a Tareas y Habilidades. En su lugar, se ha de determinar cómo comprobar la capacidad del solicitante en función de lo que haya hecho y pretenda hacer en el Proyecto.[31]

Una de las primeras Desarrolladoras no empaquetadoras me señalaba en una entrevista la importancia de esta Resolución General:[32]

> **MadameZou:** Yes, because there are a lot of people that work on Debian, doing various things, and they need to be full part of Debian, with the right to vote. It's a great idea.

[30] Que empieza en «GR: welcome non-packaging contributors as Debian project members», en Debian-vote (2010c). Se puede ver un resumen de los principales temas discutidos en Debian-vote (2010d).

[31] Indicamos aquí algunos lugares donde consultar algunas modificaciones posteriores relativas al NMP, que lo mejoran y simplifican: Debian-devel-announce (2011, 2012b, 2016b,a, 2017).

[32] Se puede consultar el informe de su *Application Manager* en Debian-newmaint (2011b). Y se puede leer una entrevista interesante al respecto en Hertzog (2012b).

Lo importante no son pues tanto los privilegios técnicos como la posibilidad de participar y ser parte de la comunidad:

> **MadameZou:** To he honest, I didn't need to be a Debian Developer, because I do no packaging work, I didn't need access to Debian machines. But people with whom I work started to say to me, ok, now it's time, you deserve it, you are ready [...]. If people that work with me say that it is time, ok, it's time, I need to do it. And it's great because you can participate really in the decisions.

Este reconocimiento de la colaboración desde diferentes ámbitos se reforzó en 2012 con la Declaración de Diversidad del Proyecto Debian, impulsada precisamente por esta Desarrolladora. La Declaración de Diversidad de Debian fue aprobada (también por una mayoría muy amplia) por Resolución General en 2012 (Debian.org/vote, 2012) y publicada en la *web* de Debian (Debian.org, s.f.[ab]). El texto aprobado establece lo siguiente:

> **Diversity Statement**
>
> The Debian Project welcomes and encourages participation by everyone.
>
> No matter how you identify yourself or how others perceive you: we welcome you. We welcome contributions from everyone as long as they interact constructively with our community.
>
> While much of the work for our project is technical in nature, we value and encourage contributions from those with expertise in other areas, and welcome them into our community.

Esta Declaración recoge dos hilos: el que estamos analizando referido a la diversidad de formas de colaboración que son necesarias en Debian, y el reconocimiento de la diversidad social y personal entre los miembros del Proyecto. Este segundo aspecto no lo analizaremos aquí, pero en cuanto a su importancia y su reconocimiento en Debian ha sido muy importante la creación, dentro del Proyecto

Debian, del subproyecto *Debian Women* (Debian.org, s.f.[ac]), iniciado en 2004. Así, en la página *web* «About Debian Women» (Debian.org, s.f.[ad]) se puede leer:

> The Debian Women project was founded in May 2004. We seek to balance and diversify the Debian Project by actively engaging with interested women and encouraging them to become more involved with Debian. We will promote women's involvement in Debian by increasing the visibility of active women, providing mentoring and role models, and creating opportunities for collaboration with new and current members of the Debian Project. We welcome the involvement of all people who are interested in increasing the participation of women in Debian.

Por requerimiento de la Constitución de Debian (Debian-project, 2012a), el texto de la Declaración de Diversidad debió ser aprobado formalmente por una Resolución General (por ser una Declaración de Posición del Proyecto), aunque ya se había alcanzado un consenso suficiente en la larga discusión que se produjo en la lista `debian-project`,[33] donde se reformuló a partir de diferentes sugerencias sobre el borrador original, y donde se discutió sobre el significado de la diversidad y la discriminación.

El texto publicado no establece obligaciones concretas, no es un código de conducta. De lo que se trata, según el consenso alcanzado en la discusión, es de hacer público y explícito que cualquiera es bienvenido al Proyecto y que todas las aportaciones son importantes. Aunque no supone de ningún modo alguna modificación respecto al NMP, esta declaración está en línea con la Resolución General de 2010 sobre la aceptación de Desarrolladores no empaquetadores, en tanto en ambas se establece que todas las contribuciones, incluidas las no técnicas, son importantes. Y sobre todo con la apertura propia del Proyecto y una de sus consecuencias, la obligación de recibir que discutimos en la sección 5.3, *La obligación de recibir*.

[33] A partir del mensaje Debian-project (2012b).

6.3.3.2 Formas de reconocimiento de la colaboración y de la diversidad.

En consonancia con los esfuerzos vistos por reconocer propiamente todas las formas de colaboración, se han producido también en la historia de Debian algunas otras iniciativas con el objetivo de dar más visibilidad a aquellos que contribuyen al Proyecto sin ocupar una posición oficial (Desarrollador de Debian o Mantenedor de Debian), sobre todo de los que empiezan a colaborar, independientemente de si se hace empaquetando *software* o de cualquier otra forma. Seguimos por tanto en la tensión entre los dos sentidos de comunidad (en cuanto a su extensión) que veíamos, aunque no estemos ya propiamente en la distinción técnico / no técnico.

Dentro de estas formas de reconocimiento, una de las más importantes es la creación del sitio *web* «Debian Contributors» (Contributors.debian.org, s.f.[a]), una iniciativa del Desarrollador de Debian Enrico Zini. El origen lo podemos encontrar en un *post* de su *blog* elaborado junto a la Desarrolladora Francesca Ciceri (MadameZou), «More diversity in Debian skills» (Zini y Ciceri, 2012):[34]

In his Debconf talk, zack said:

> We need to understand how to invite people with different backgrounds than packaging to join the Debian project [...] I don't know what exactly, but we need to do more to attract those kinds of people.

Francesca and I know what we could do: **make other kinds of contributions visible**.

Esta visibilidad permitirá el reconocimiento de todas las contribuciones, lo que a su vez se espera que aumente esas contribuciones. En ese *post* se indican también algunas fuentes de las que obtener los datos de participación.

[34] Se puede encontrar una iniciativa parecida, vinculada con la anterior, en «Debian whois: a proposal to increase visibility and recognition of contributors» (Ferrari, 2012).

Un año más tarde, a partir de DebConf 13 (Zini, 2013b), se pone en marcha esta iniciativa. Es oficialmente anunciada en la lista `debian-devel-announce` (Debian-devel-announce, 2013), donde se anuncia el uso del nombre *Debian Contributor* o Colaborador de Debian para referirse a cualquiera que colabore, de cualquier forma, con Debian. A diferencia de las figuras de Desarrollador de Debian o Mantenedor de Debian, no implican ningún proceso oficial para ser reconocido como tal. Se trata tan sólo de recoger en la página antes mencionada el nombre y las colaboraciones, como una manera de dar crédito a todos los que hayan colaborado. El proceso de recogida de datos es diverso y necesariamente incompleto, dada la propia diversidad del Proyecto y de las formas de colaboración en el mismo, y los diferentes equipos o Desarrolladores pueden añadir nuevas fuentes de datos.[35]

En septiembre del 2013, el impulsor de esta iniciativa publica un breve *post* en su *blog* que expresa y condensa algunas de las ideas desarrolladas en este trabajo de investigación (Zini, 2013a):

> **A moving commit**
>
> A moving commit: an entire class of people who previously did not have a name, now have a name.
>
> I feel that this simple patch, although it is just a little first step, is a significant moment in the history of Debian.

La expresión «A moving commit» es un enlace[36] al *commit* (véanse las secciones 3.2.5, *Doocracia y agencia: la mediación técnica*, y 4.8, *Repositorios de software libre*) que implementa la nueva categoría «Debian Contributor» en el código de la página «Debian New Members» (Nm.debian.org, s.f.[a]), haciéndolo así efectivo. Es un cambio que toma cuerpo como un pequeño parche (véase la sección 2.2.6, *De las cajas negras a los parches como metáfora de lo técnico*), pero que efectivamente se puede considerar un momento importante en la historia de Debian. Quizás suponga también, forzando un poco la traducción y la

[35] Véase al respecto Contributors.debian.org (s.f.[b]). También Debian-devel-announce (2014).
[36] En el momento de escribir esto, el enlace que hay en el *blog* no funciona. Ahora debería apuntar a Salsa.debian.org (2013b).

relación, *commit to an interpretant* (véase la sección 2.2.8, *Agencia y semiótica*), comprometerse con un interpretante: si tomamos el parche y la categoría «Colaborador de Debian» como signo y la comunidad como objeto, podríamos ver aquí un interpretante que expresa la necesidad de constituir esa comunidad a partir de cualquiera que contribuya al Proyecto, no sólo de los miembros oficiales del Proyecto.

6.3.4 Algunas conclusiones sobre diversidad y procesos de reforma.

Respecto a la evolución y las reformas que hemos visto sobre el NMP, mi interpretación es que, aunque hasta la propuesta de 2008 relativa al «Developer Status» con la que empezábamos este análisis, esta evolución se puede describir efectivamente como una crisis de crecimiento y una crisis ética en tanto que de lo que se trataba fundamentalmente era de construir y asegurar la confianza (Coleman y Hill, 2005; Sadowski *et al.* 2008; Wallach *et al.* 2005; Zacchiroli, 2012b), la evolución posterior supone fundamentalmente el intento de adecuar la pertenencia formal al Proyecto con la idea o ideas de comunidad que funcionan en Debian. Lo que encontramos en las propuestas de reforma del proceso de pertenencia es precisamente el intento de actualizar, de materializar, las nociones de «comunidad» que estructuran las relaciones dentro del Proyecto Debian. Se trata de entender en su interrelación las ideas sobre la comunidad que ponen en juego los miembros de Debian con las prácticas sociales de las que surgen, y a las que intentan dar forma. Por eso intentamos construir etnográficamente el concepto de comunidad en Debian en sus aspectos *emic*, determinando qué prácticas sociales dan cuerpo a esa conceptualización, y a su vez cómo se realiza y actualiza esa noción o nociones en las prácticas sociales, sin confundirlas.[37]

Una noción de comunidad que, como vimos, ha de ser igualitaria, no jerárquica, y no establecer clases diferentes de miembros, más allá de las relacionadas con la doocracia y técnicamente necesarias. Estas diferencias respecto a accesos y privilegios técnicos, y por lo tanto relativas a la doocracia, no deberían suponer diferencia social, respecto al voto, en relación a la pertenencia a la comunidad.[38]

[37] Sobre esta cuestión puede verse Amit (2002).
[38] Respecto a esta relación entre lo técnico y la doocracia por un lado, y lo social y la democracia por otro, véase el capítulo 2, *Debian como trama sociotécnica*.

De ahí que a lo largo de todo este proceso de reformas, desde la creación de la figura del *Debian Maintainer*, se haya producido *de facto* un proceso de fragmentación de lo que significa ser miembro de Debian, en base a una modularización de los derechos y privilegios que supone esa pertenencia. Y paralelamente, del NMP. Una consecuencia de ello es que se muestra claramente que el derecho que define la pertenencia es el de voto, no el de subir paquetes al archivo, en tanto es el que comparten todos los Desarrolladores. También hay que tener en cuenta las resistencias a esta adecuación que se manifiestan en un cierto elitismo, o la importancia concedida en algunos debates a la confianza relativa a la capacitación técnica por encima de la relativa a los principios expresados en el Contrato Social o las DFSG.

El elemento más importante en estas discusiones y procesos de reforma es el papel que juega la colaboración entendida como don en relación a la pertenencia a la comunidad (véase el capítulo 5, *El don del* software *libre*). La pertenencia a la comunidad debía ser reconocida de un modo u otro, en mayor o menor medida, a todo aquél que contribuyera al Proyecto, que debe ser reconocido precisamente *porque* contribuye. Lo importante aquí es que eso produce por sí mismo, de alguna manera, pertenencia a la comunidad, de manera similar al funcionamiento de la doocracia. La cuestión es sobre todo, en las discusiones sobre la reforma de los procedimientos considerados, hasta qué punto (es decir, con qué privilegios) esta pertenencia a la comunidad ha de ser formalmente reconocida. En 2009, quien proponía los cambios en el NMP en el anuncio sobre «Developer Status» que hemos discutido, decía:

> **Fernando:** Why do you think Debian needs more levels of members?
>
> **ganneff:** Because our current system of entry is all or nothing. You can be a Debian Developer or you are nothing. Of course people do recognize others doing work like those doing translations, but it is people that recognize them, not the project. We happily take translations, we happily take people maintaining our website, [...] but in no way we are giving back anything to them. [...] It would be nice to have a payoff, recognizing them as being part of Debian.

El otro elemento relevante sería la diversidad como seña característica de la comunidad Debian. En numerosas teorizaciones antropológicas[39] sobre la comunidad se otorga una gran importancia a las ideas de identidad e identidad colectiva, relacionando la interpretación de la noción de comunidad por parte de un grupo de personas con los procesos de identificación y categorización a partir de los que se piensan y construyen su identidad como miembros de un grupo social. Pero la noción de comunidad que encontramos en Debian pasa más por la diversidad que por la idea de identidad. Los miembros de Debian no se sienten parte de una comunidad porque se sientan parecidos unos a otros (desde luego, sin prejuzgar ni entrar en la cuestión de hasta qué punto comparten elementos en común). De hecho, insisten continuamente en que es precisamente la heterogeneidad interna lo que les caracteriza como grupo:[40]

> **ana:** [Tras citar una serie de estudios y profesiones de miembros de Debian que no tienen nada que ver con la informática, y que incluyen filología, anestesista, matemáticas, psicología, administración de empresas o ingeniería de caminos] Yo creo que es una de las cosas clave para entender Debian [...] lo primero de todo de Debian es que hay de todo. [más adelante indica también numerosas diferencias en relación a aficiones, religión, opciones políticas, nacionalidades o lenguas.]

> **dato:** Siempre he dicho en las charlas que el mayor reto de Debian es su tamaño, y luego su heterogeneidad, en cuanto a Desarrolladores. Somos un grupo muy muy diverso, de orígenes diversos, de creencias y opiniones diversas. Entonces, estamos aquí todos para lo mismo, hacer una distribución de *software* libre que sea la mejor. Pero incluso ciñéndonos al ámbito del *software* libre tenemos diferencias muy importantes entre nosotros, como el pasado atestigua, cada vez que he-

[39] Véanse por ejemplo Barth (1976) y Cohen (1985). Brubaker y Cooper (2000) proponen un vocabulario más rico y matizado que el de la simple identidad: identificación y categorización; autocomprensión; comunalidad, conectividad, agrupación.

[40] Y también a la distribución a nivel técnico: como vimos, Debian se caracteriza como «El Sistema Operativo Universal», al ser instalable en un gran número de arquitecturas diferentes, y al pretender empaquetar la mayor cantidad de *software* posible (véase la sección 4.1, *La Red como metáfora. Unix*).

> mos discutido sobre el *firmware* en el *kernel*, o de las licencias, o de tal [...] También diferencias culturales o de maneras de ser.

En el Proyecto Debian encontramos entonces una distancia entre, por un lado, el alto nivel de esfuerzo y dedicación que puede llegar a suponer y la fuerza de los lazos y sentimientos que genera, y, por otro lado, la falta de «identificación», la ironía, el desapego, la individualidad, la conciencia de la diferencia con otros miembros del Proyecto. No se enfatiza la homogeneidad (o se limita mucho su alcance), ni ésta incita a la unidad de acción o pensamiento.

6.3.5 Un colectivo de humanos y no humanos.

En la primera sección de este capítulo se propuso considerar a Debian como un colectivo sociotécnico, que se sostiene por la asociación de elementos humanos y no humanos, técnicos. A lo largo de todo este trabajo hemos ido describiendo (parte de) la trama que constituyen conjuntamente, y hemos comprobado que no se puede entender la práctica social de los primeros sin una atención detallada a las características y la agencia de los segundos. Una consecuencia de todo esto es que la pregunta sobre quién es miembro de Debian no puede ser sólo una pregunta sobre qué humanos son parte del mismo, sino que debería extenderse hasta preguntar qué objetos son parte de Debian (incluyendo los paquetes en el archivo, la infraestructura, o los medios de interacción). Y de hecho, dos de los principales problemas de Debian a ojos de muchos de sus Desarrolladores se refieren precisamente, como hemos dicho ya, a la aceptación e integración de nuevos miembros, humanos (nuevos Desarrolladores) y no humanos (paquetes en el archivo) en Debian. Éstas son discusiones recurrentes en los canales de comunicación del Proyecto Debian. Sobre esta relación se puede mencionar por ejemplo un mensaje de correo a la lista `debian-project` en 2009, con el asunto «DAM and NEW queues processing» (sólo se cita una parte) (Debian-project, 2009j):[41]

```
To: debian-project@lists.debian.org
Subject: DAM and NEW queues processing
From: Lucas Nussbaum <lucas@lucas-nussbaum.net>
```

[41] Sobre DAM, véase el cuadro 6.1, *Terminología sobre miembros, mantenedores y desarrolladores*; sobre NEW, véase la sección 5.2.3, *Las licencias en Debian*.

> Date: Tue, 23 Jun 2009 11:30:53 +0200
>
> For years, the DAM and NEW queues have been the major source of frustration in the Debian community. Several attempts have been made to improve the situation, but the problems have never been really solved so far. And the queues are again in a bad state.
>
> First, DAM: […]
>
> Then NEW. […]
>
> It is clear, based on the previous attempts to solve those problems, that simply throwing more manpower on the DAM and ftpmasters team won't solve those problems. We have tried that for years, and it has failed for years.
>
> We need to compromise on the level of quality we expect from our prospective DDs and new packages. We need to accept that new maintainers are humans, and will sometimes make mistakes, no matter how many questions you ask them. And that packages with licensing problems will continue to be uploaded to Debian despite all the checking done by ftpmasters: licence problems are already being introduced in packages that already went through NEW (through the addition of new code in existing packages, for example). We put a lot of energy into processes that are not totally efficient.

A este correo le sigue una larga discusión, que es también muy interesante para profundizar en el funcionamiento práctico del NMP y la revisión de licencias en NEW, el lugar donde, entre otras cosas, se revisan las licencias de los paquetes subidos antes de ser integrados en el archivo de Debian. No muchos Desarrolladores están de acuerdo, en esa discusión, con las soluciones propuestas. Lo que considero interesante del correo es precisamente que esos dos problemas (el NMP y el funcionamiento de NEW) aparecen estrechamente relacionados. Esto no es casual. El *software* libre en general, no sólo Debian, se caracteriza por la invención de dispositivos para incluir a los usuarios en la comunidad, de los que hemos visto algunos: el BTS, las listas de correo, los repositorios de código, etc. El NMP es también uno de estos dispositivos. Todos, incluyendo las reformas del NMP, tienen en común que son formas de implementar la obligación de recibir (véase la sección 5.3, *La obligación de recibir*). De lo que se trata es de facilitar el reclutamiento y la colaboración. Esta íntima relación entre sujetos y objetos, humanos y no humanos, es además un rasgo definitorio de la noción de colectivo: «Los objetos y los sujetos se fabrican simultáneamente, y un incremento en el número de sujetos es

algo directamente relacionado con el número de objetos suscitados –elaborados– en el colectivo» (Latour, 2001, pág. 234). En un colectivo sociotécnico, los actores (y sus objetivos) se transforman mutuamente.

En la consideración de Debian como colectivo destaca la cuestión de la escala del colectivo, en tanto que recluta muchos no humanos (algunos de ellos muy lejos), lo que lo convierte en un colectivo moderno (Latour, 2001, pág. 234):

> la diferencia [entre colectivos «primitivos» y modernos] estriba en que el segundo colectivo traduce, entrecruza, enrola y moviliza más elementos, y en que esos elementos se hallan más íntimamente conectados, más finamente entretejidos en la tramazón social que los elementos del primero. Es crucial la relación entre la escala de los diferentes colectivos y el número de no humanos alistados en su seno.

Respecto a Debian, una de las preguntas que nos hemos formulado es cómo se puede mantener un colectivo tan grande pero sobre todo tan espacial y temporalmente disperso. En buena parte, esto es posible gracias todos los elementos que ha sido capaz de enrolar y reclutar. Es la estrecha asociación entre humanos y no humanos lo que permite esta estabilidad. Una consecuencia de ello es, que como ya hemos visto, la distinción entre lo técnico y lo social (véase el capítulo 2, *Debian como trama sociotécnica*) se produce siempre *a posteriori* en un colectivo, no puede ser algo dado de antemano: «el colectivo moderno es un colectivo en el que las relaciones entre los humanos y los no humanos son tan íntimas, en el que hay tantas transacciones y tan alambicadas mediaciones, que no hay ningún sentido plausible en el que puedan distinguirse los artefactos, las corporaciones y los sujetos» (Latour, 2001, pág. 235). O, como vimos, sólo puede producirse después de un trabajo de purificación.

Pero si Debian se entiende mejor como un colectivo de humanos y no humanos, es también porque se basa en la doocracia (véase la sección 3.2, *Toma de decisiones: de la meritocracia a la doocracia*). Como vimos, ésta se cimenta en las asociaciones entre humanos y dispositivos técnicos. Si alguien puede decidir en base a la doocracia sobre determinada área, normalmente es a través de los

dispositivos o elementos técnicos que controla. En Debian encontramos una relación significativa entre los derechos y los privilegios que se tienen, y el acceso técnico a los recursos. Como explica un Desarrollador de Debian, no son idénticos (Debian-devel, 2012e):

> To: debian-devel@lists.debian.org
> Subject: Maintainers, teams, Uploaders, DMs, DDs, etc.
> From: Ian Jackson <ijackson@chiark.greenend.org.uk>
> Date: Wed, 13 Jun 2012 16:35:12 +0100
>
> […]
> But firstly I want to explain an important principle: the distinction between ability and permission. In any complex collective endeavour, there will be both access controls that actually prevent some people from taking certain acts, and ideas (whether stated formally and clearly, or informally and vaguely) about what should be done when and by whom. Access restrictions
>
> are annoying to implement and work with so they should be minimised where possible - but they are a necessary backstop to prevent abuse. […] So for
>
> example, DDs have enormous theoretical power but there are strong and well documented social controls on how they should exercise that.

Pero sí que tienen una estrecha relación, y se piensan unos en función de los otros. Como explicaba un Desarrollador de Debian en una entrevista, se pueden relacionar los niveles de privilegio con la filosofía UNIX, incluso se puede comparar el Proyecto con una máquina UNIX (véase la sección 4.1, *La Red como metáfora. Unix*):

> **mones:** [En relación a las reformas del NMP que ya hemos visto] Un proyecto de este tamaño siempre necesita gente, restringir la capacidad de que la gente haga cosas no es necesariamente bueno. Además va totalmente en contra de la filosofía de los entornos UNIX, que es siempre un entorno colaborativo. [En UNIX] el tema de los permisos, de los grupos, se inventó para que la gente pueda colaborar en las mismas áreas. Lo veo en la misma línea filosófica.

> Hasta ahora tener *Debian Developers* significaba que sólo tienes un grupo en la máquina. Pero ninguna máquina tiene sólo un grupo de usuarios. Y realmente ya existe, porque no todos los *Developers* tienen acceso a todos los recursos del proyecto. En realidad ya hay varias categorías, que no están explicitadas pero las hay. O sea yo no tengo acceso a rechazar un paquete de la cola NEW, y la gente del FTP Team sí tiene [...] Realmente ya hay muchos grupos con diferentes niveles de privilegios. No por ser *Developer* de repente se abre un mundo de privilegios y puedes hacer lo que te de la gana en el Proyecto, no, ni de lejos.

De hecho, una manera de determinar quién es Desarrollador de Debian (y no Debian Maintainer o alguien que tiene una cuenta de invitado) es a través de los permisos UNIX que se tienen en diferentes máquinas.[42] Por otra parte, determinar las capacidades técnicas a través de este sistema es poco flexible y tiene sus propios problemas. En un *post* en su *blog* escribía el Desarrollador de Debian Joey Hess (2006): «In Debian, we work on developing a unix-style operating system, and when the project was getting started, the obvious way to deal with delegating powers to various people was naturally to use standard unix permissions on our master server», una situación con la que se muestra crítico y que pretende que se vaya transformando para conseguir, precisamente, una mayor apertura:

> Anyway, the point of this is that, if you survey the parts of dealing with the project where Debian developers feel most helpless and unempowered, the parts that are over and over again the subject of heated discussions and complaints, you will find that those are the parts of the project where unix permissions still hold sway. [...] The challenge, then is to find ways to open things up to everyone, without throwing security out the window.

Una mayor apertura, decíamos, porque el prerrequisito de la doocracia es que el acceso a los dispositivos en torno a cuyo acceso se construye esté lo más

[42] Puede verse el mensaje de correo a la lista `debian-devel` Debian-devel (2014e), con el *subject* «Re: "Determining, ad hoc, whether someone is a DD"».

abierto posible, permitiendo que se pueda efectivamente crear una asociación con ellos. Si los permisos UNIX son una manera de controlarlos, es normal que ahí se concentren esas partes problemáticas del Proyecto.

Otra de las formas de determinar si alguien es Desarrollador o Mantenedor de Debian es a través de los diferentes *keyrings* en los que se puede encontrar su clave criptográfica GPG (Debian-devel, 2014f). Esto nos lleva directamente a la última sección de este capítulo.

6.4 La construcción tecnosocial de la confianza.

Hemos visto que uno de los objetivos fundamental del *New Member Process* es la generación de confianza, que el Proyecto Debian y sus Desarrolladores puedan confiar en el Nuevo Desarrollador. Como ya hemos visto a lo largo de toda la investigación, en el caso de una distribución de *software* esto es especialmente importante. La razón la expresa con mucha claridad por ejemplo O'Neil:

> [c]ontributors to Debian really do have the potential to harm the software. As a result, [...] contributions cannot be anonymous. Online communities are routinely described as having fluid boundaries and shifting members and identities. In contrast, members of FOSS projects who are developing software that is hosted on protected servers connected to the Internet must maintain a distinct and trusted identity, which will enable them to gain access to these protected resources (O'Neil, 2009, pág. 135).

En el caso de Debian, entonces, la confianza y la identidad son tópicos inseparables. Una consideración sustantiva del significado y las formas de confianza excede con mucho las pretensiones y los límites de la presente investigación. Una introducción a las cuestiones que plantea esta noción puede encontrarse en Velasco, Díaz de Rada *et al.* (2006). Aquí, nos quedamos con la siguiente consideración genérica sobre la confianza:

> Es verdad que, en último término, para legos y expertos, [...] el sentido dominante de la confianza es el que remite a las relaciones interpersonales cara a cara y evoca interacciones significativas de larga duración. Esa forma de confianza, basada en la capacidad de los seres humanos –bien identificada por Schutz, Goffman y otros microsociólogos– para generar consensos, expectativas de rol y convenciones compartidas a partir de las rutinas de interacción emergentes en encuentros repetidos, ubica la expectativa fiduciaria en un plano personal: el de sujetos que se conocen unos a otros por experiencia práctica, inmediata, a lo largo del tiempo (Velasco, Díaz de Rada *et al.* 2006, pág. 26).

Ya vimos en la sección 5.3.1, *El crecimiento del procomún y la obligación de recibir*, la estrecha relación que se da entre copresencialidad y confianza. Dado que hablamos de un colectivo que interactúa fundamentalmente a través de medios electrónicos y dispositivos tecnológicos, ¿pueden ser suficientes estas interacciones para generar confianza?[43] Una confianza densa o interpersonal, más allá de la confianza figurada, según la distinción que se establece en Velasco, Díaz de Rada *et al.* (2006, pp. 312 ss.). Si ésta es más bien una forma de cooperación que se da en contextos institucionales e impersonales, aquélla «exige la definición de sujetos, situaciones y tiempos concretos» (Velasco, Díaz de Rada *et al.* 2006, pág. 316).

Como hemos dicho, uno de los lugares fundamentales donde se generan precisamente «consensos, expectativas de rol y convenciones compartidas», es decir confianza, es precisamente el *New Member Process*. La confianza que se desarrolla en este proceso se puede entender como una construcción tecnosocial en al menos dos sentidos. Por una parte, y como ya vimos en la sección 6.3.1.2, *Mentorización, confianza, y doocracia*, en relación a aquello sobre lo que se debe confiar: la competencia técnica, en la parte del NMP llamada «Tareas y Habilidades», pero también la aceptación de los objetivos, los principios y la filosofía del Proyecto, en la parte llamada «Filosofía y Procedimientos», que haría referencia a la parte «social» según la oposición habitual en Debian. Por otra parte, la confianza también es tecnosocial respecto a la forma de comprobarla y asegurarla, especialmente en

[43] Sobre la confianza y las relaciones en entornos *online*, desde una perspectiva fundamentalmente filosófica, puede verse Ess y Thorseth (2011).

su conexión con la identidad. La forma de comprobar ésta es diversa, local y contingente, en un proceso en el que es imposible separar los aspectos que en Debian son denominados «técnicos» y sociales». Mackenzie lo expresa con claridad en el contexto de un estudio sobre la automatización de la prueba de teoremas matemáticos:

> the sociotechnical processes underpinning the modest effectiveness of review and testing are local. They involve the generation of trust in particular, personally known people [...] The interweaving of knowledge of things and knowledge of people can be expected to be effective only in situations where there is sufficient continuity of personnel (MacKenzie, 2004, págs. 304-305).

En conjunto, se están usando al menos tres nociones de confianza: en la capacidad y competencia técnicas, en el acuerdo sobre formas de entender el *software* libre, y en que se pueda determinar la identidad de los implicados en caso necesario (porque se tengan intenciones ocultas que puedan poner en peligro al Proyecto). Estas tres formas de confianza son además esenciales para que pueda funcionar la doocracia como forma de gobierno. Ésta sería imposible sin el establecimiento de un cierto nivel de confianza en que se mantendrán las expectativas de comportamiento de los diferentes actores implicados.

Una cuestión relativa a la confianza que no abordamos, pero que mencionamos muy brevemente, es el establecimiento de la confianza, por parte del usuario, en la distribución Debian o en el *software* libre en general. Un Desarrollador hablaba de la confianza que permite el *software* libre en los siguientes términos:

> **dato:** Desde hace unos años cada vez dependemos más de los ordenadores. Les confiamos nuestras fotos, nuestros vídeos, nuestros correos electrónicos, nuestras intimidades, trabajos de universidad, nuestra salud, se lo confiamos todo [...] Entonces considero muy importante que esa confianza no sea ciega [...] o sea, no nos encontremos el día de mañana con que el ordenador te ha traicionado, que el programa que

> utilizabas para hacer tu tesis te dice que si no pagas tanto ya no te deja editarla más. No sólo por el pagar, sino por la libertad inherente [...] Que el código sea abierto es una gran garantía de que pase lo que pase, tú puedes seguir, tienes tus datos o lo que sea.

En el *software* libre esa confianza quedaría garantizada por la apertura del código y las licencias, y en Debian en particular por la apertura de sus procesos y sus comunicaciones. Nos centraremos a partir de aquí en el papel de las claves criptográficas GPG, tanto en el NMP como en el resto de prácticas sociales del Proyecto Debian, como un lugar privilegiado donde se vinculan los aspectos sociales y los técnicos. Este sistema criptográfico supone además un modelo de valoración de la identidad (*identity assessment*), un modelo de confianza (Wolf y Quiroga, 2018, pág. 2).

6.4.1 Identidad y claves GPG.

GPG (Gnupg.org, s.f.[a]) es un *software* criptográfico, una implementación libre del estándar OpenPGP, un sistema de criptografía de clave pública.[44] En este tipo de sistemas, cada usuario posee dos claves criptográficas, una pública y otra privada. La clave pública es accesible a todo el mundo, a menudo a través de un servidor de claves, y permite a cualquiera enviar un mensaje cifrado a su dueño, un mensaje que sólo puede ser descifrado por medio de la clave privada. La clave privada sólo es accesible por su dueño (además, necesita ser desbloqueada por una contraseña), y permite firmar documentos como correos electrónicos o paquetes de *software*. Usando la clave pública correspondiente se puede verificar que la firma ha sido efectivamente realizada con una clave privada determinada. Permite también comprobar que lo que se ha firmado no ha sido modificado después de ser firmado. El sistema permite así no sólo cifrar las comunicaciones digitales, sino asegurar que determinado documento o paquete ha sido generado por el poseedor de una determinada clave privada. Las tres funciones básicas de un sistema de este tipo serían entonces el cifrado, la firma y la atribución de identidad (Wolf y Gallegos-García, 2017, pág. 1). Ya hemos visto algún ejemplo de cómo funciona.

[44] Se puede encontrar información técnica sobre estos sistemas en Riseup.net (s.f.) o Gnupg.org (s.f.[b]). Una buena explicación técnica general y un recorrido por su historia (y de la criptografía en general) puede encontrarse en Wolf (2016).

En el correo que aparece en la página 60, la expresión «PGP Signed Part:Good signature from A2D13001D98C0FBA Don Armstrong <don@debian.org> (trust full) created at 2012-10-05T22:53:13+0200 using RSA» es una comprobación que hizo mi cliente de correo, mostrando que el correo está firmado efectivamente por ese Desarrollador. Mi cliente de correo muestra eso porque firmé esa clave con la mía a partir de un encuentro presencial (enseguida veremos qué supone esto). En el mensaje que aparece en la página 72 se puede ver al final que aparecen las firmas de los que lo mandan como documentos adjuntos. A partir de esos documentos es posible determinar si son ellos quienes han enviado efectivamente el documento.

El problema que surge entonces es el de conectar una determinada clave privada con una persona. Cada usuario crea sus propias claves, y podría asignarles cualquier nombre de persona y dirección de correo electrónico. El sistema criptográfico es capaz de vincular por sí mismo cada conjunto de datos firmado con una clave, pero no a ésta con un individuo concreto. Este vínculo ha de establecerse, en principio, personalmente. Si te conozco y me das personalmente tu clave pública, puedo estar razonablemente seguro de tus firmas GPG. Por otra parte, una de las cosas que se pueden firmar con una clave privada es, precisamente, la clave pública de otra persona. Esto permite construir lo que se conoce como Anillo de Confianza o *Web of Trust*, un conjunto de claves relacionadas por las firmas de unas a otras. Si accedo a una clave pública puedo determinar si es la que se ha usado para firmar un determinado conjunto de datos, pero no puedo en principio conectarla a una identidad personal. Pero si esa clave está firmada por otras claves en las que he confiado previamente (más adelante veremos cómo), puedo establecer un nivel de confianza respecto a que se corresponde con determinada identidad personal. Esta confianza aparece así como transitiva: se confía en que la identidad está bien establecida si confían en ella aquéllos en los que yo confío. Y la identidad de la que hablamos se refiere entonces a una operación de identificación muy concreta, el conjunto de procesos mediante los que los miembros de Debian conectan los actos de un participante con una determinada persona física. Como señalaba un Desarrollador de Debian en una entrevista, esto produce una cierta definición y determinación colectiva de la identidad:

luciano: De alguna forma funciona como el concepto colectivo de ...,

> quien crees que sós hace que seas alguien. Entonces, quien te define es el colectivo y no uno, no las individualidades, creo yo. De alguna forma, lo que se llama la *web of trust*, la telaraña de confianza, es eso.

Por otra parte, esto puede convertirse también en uno de los problemas de este modelo. Así, algunos activistas piensan que puede dar demasiada información sobre las relaciones sociales que se mantienen:

> **micah:** In the political work [...], lots of people [...] often want to use something like GPG to communicate securely. But they are not comfortable with the web of trust, because they don't want to disclose to people their social relationships. If we sign keys and then we upload them to the keyserver, then anyone can look and see that we have exchanged keys, so we have met in person. [...] Many activists don't want that information to be known. They don't want to have this kind of surveillance. So they have this conflict, they want to use GPG for the encryption capabilities, but the identity part is slightly problematic.

El anillo de confianza es uno de los dos modelos básicos de confianza que se pueden dar en un sistema de criptografía de clave pública, siendo el otro dependiente de autoridades centrales de certificación. Este es el modelo centralizado que se aplica por ejemplo en el uso de los protocolos TLS y SSL (véase la página 206). El modelo del anillo de confianza es descentralizado y distribuido, y se plasma en el anillo de llaves o *keyring* que está a disposición de los usuarios del sistema. En el caso de Debian, y como es caracterizado por parte de uno de los responsables de mantener estos anillos en Debian (Wolf y Gallegos-García, 2017; Wolf y Quiroga, 2018), se trata de un anillo de confianza *curado*, puesto que existe un equipo que mantiene y actualiza los anillos de claves del Proyecto (Wiki.debian.org, s.f.[aw]). Este trabajo es esencial, en tanto que los derechos de voto en las Resoluciones Generales, y de subir paquetes al archivo de Debian, sólo se pueden ejercer mediante el uso de una clave criptográfica. Lo que este equipo mantiene son los anillos efectivamente usados en el Proyecto, y que se pueden descargar como paquetes, pero a través de las firmas cruzadas con usuarios que no están en esos anillos, son parte de la *web of trust* que incluye todas las firmas GPG conectadas.

Así pues, como vimos, el primer paso en el proceso para convertirse en Desarrollador de Debian es integrarse en la *web of trust*.[45] Sólo al final del proceso se añadirá la clave a uno de los llaveros mantenidos por el Proyecto, pero tener una clave firmada por al menos dos desarrolladores permite que los miembros de Debian puedan evaluar la identidad del sujeto y relacionar sus actividades con esa identidad.

6.4.2 Firmado de claves: identidad y confianza.

El firmado de una clave de otra persona es un proceso que ha de realizarse *offline*, en un encuentro cara a cara. Así, para iniciar el *New Maintainer Process* es necesario encontrarse personalmente con algunos Desarrolladores. Pero el momento en que se producen la mayor parte de estas firmas cruzadas que constituyen el anillo de confianza son las llamada Fiestas de Firmado de Claves o *Keysigning Parties*.

Una Fiesta de Firmado de Claves consiste en un intercambio de firmas en las claves personales de los participantes, en el que participan varios o muchos individuos (Wiki.debian.org, s.f.[ax]). Veamos los pasos que se siguen, tanto en una *Keysigning Party* como un encuentro entre dos personas. En primer lugar se obtienen las diferentes claves públicas impresas en papel. En realidad, dado el tamaño de las mismas, lo que se intercambia es una huella digital o *fingerprint* (producida a partir de la clave y mucho más pequeña) de la misma, junto con un nombre y una o varias direcciones de correo electrónico. A partir de esa huella puede obtenerse la clave completa desde un servidor de claves. En una Fiesta de Firmado de Claves como las que se producen en las DebConfs, se produce un documento que las reúne a todas. En este caso, cada participante debe asegurarse de que el documento es el correcto y no ha sido alterado, gracias de nuevo a una huella criptográfica del documento. De una u otra manera, esto asegura que la clave que se firma es la correcta.

En segundo lugar, se toma la decisión de firmar o no una determinada clave con la propia. Veremos en seguida que existen diferentes criterios respecto a esta decisión, pero lo típico suele ser comprobar algún documento oficial que permita

[45] Véase también Coleman (2005, 378 ss.).

establecer una correspondencia entre el nombre que aparece junto al *fingerprint* de la clave y la identidad *offline* del interesado. Firmar la clave de otra persona significa así certificar, para sí mismo y para otros, que la clave que se firma corresponde efectivamente a un individuo que tiene oficialmente ese nombre. Dado que éste es el paso más problemático, lo discutiremos algo más detalladamente a continuación.

Una vez tomada la decisión, se firma la clave en el ordenador donde se tiene la propia clave, pues es necesario para ello la propia clave privada. Esto ocurre normalmente después de la Fiesta de Firmado, a veces unas horas después, pero normalmente días o semanas después. El procedimiento normal suele ser, después de firmarla, mandar la clave firmada al otro usuario en un correo cifrado. Se asegura así que el individuo en cuestión tiene el control efectivo tanto sobre la dirección de correo electrónico como sobre la clave privada asociada, necesaria para descifrar la firma. Una vez que la descifre, podrá enviarla a un servidor de claves público para que todo el mundo pueda ver que determinada clave ha firmado a determinada otra. Básicamente, firmar una clave significa afirmar públicamente que crees que la persona que has encontrado tiene el nombre y controla la dirección de correo que aparecen en la clave, y la parte privada de la clave.

Como vemos, este proceso de creación de confianza distribuida es un proceso tecnosocial en el que no se pueden separar los aspectos técnicos y sociales, y que por lo tanto contribuye a constituir Debian como colectivo sociotécnico. La constitución y el mantenimiento de las relaciones sociales basadas en la confianza pasa también por la mediación del sistema criptográfico. La confianza que surge del funcionamiento de la *web of trust* sólo es posible en relación a las características técnicas de ese sistema. Como vimos, la confianza remite en última instancia a relaciones interpersonales cara a cara, pero encontramos aquí un sistema que posibilita que esta confianza se construya en relación a un contexto tecnológicamente mediado y complejo, con características de sistema experto (Velasco, Díaz de Rada *et al.* 2006).

El hecho de que la firma de una clave esté necesariamente ligada a un proceso *offline* que ha de realizarse en un encuentro cara a cara supone que tiene también un aspecto de rito, como vimos en la sección 1.1, *Una visión preliminar*.

Tres escenas etnográficas, en el que se favorecen las relaciones sociales personales al tiempo que se fortalece la *web of trust*. Las Fiestas de Firmado de Claves son un acontecimiento importante en las DebConfs, dada la oportunidad que éstas suponen para encontrarse físicamente.

En las tres DebConfs en las que he participado durante el trabajo de campo correspondiente a esta investigación he participado en las correspondientes Fiestas de Firmado de Claves. Desde la de 2009 éstas se han realizado a lo largo de toda la conferencia de manera continua, no puntualmente en una reunión masiva como en las anteriores. Aunque también se han realizado (en 2009, durante el *trip day*, véase la sección 1.1, *Una visión preliminar. Tres escenas etnográficas*), se insistía en firmar las claves en encuentros más personalizados distribuidos a lo largo de la conferencia. Hay dos razones fundamentales para esto. La primera es que se consigue así potenciar una mayor interacción social entre los participantes, dejando espacio para comunicarse a propósito del intercambio de claves.

La segunda razón para la realización de una *keysigning* continua es que se permite así una comprobación con menos prisa, y por tanto más rigurosa, de la documentación pertinente. En una reunión donde cada uno de los participantes ha de comprobar decenas de documentos oficiales, a menudo emitidos por diferentes estados, es fácil que no se haga con todo el cuidado necesario. Uno de los motivos para este cambio fue la discusión causada por el experimento realizado por el Desarrollador de Debian Martin Krafft en la DebConf 6. Krafft usó en la *keysigning party* un documento expedido por la ficticia República Transnacional, que sin embargo fue aceptado por numerosos participantes, mostrando así que la comprobación no siempre se hacía rigurosamente.[46]

Pero la discusión en el proyecto[47] a partir de esta situación va mucho más allá de los problemas de basar la comprobación de la identidad en un documento oficial emitido por un estado. La cuestión de fondo más importante es la de qué es exactamente lo que se certifica al firmar una clave, qué significa ese acto. En el seno de Debian existen diferentes concepciones de lo que significa firmar una clave, y consecuentemente se mantienen diferentes requisitos para ello.

[46] Sobre esto pueden verse Krafft (2006a,b), o la discusión que sigue a Debconf-discuss (2006).
[47] Véase por ejemplo la discusión que sigue a Debconf-discuss (2009).

Si bien para todos de lo que se trata es de asegurar la identidad de la persona que maneja la clave criptográfica, esto puede significar diferentes cosas para diferentes usuarios. Si MacKenzie (2004, págs. 306 ss.) distinguía diferentes *cultures of proving*, en Debian podemos distinguir diferentes culturas de la confianza y la comprobación de la identidad. Las dos políticas básicas al respecto son, de una parte limitarse a comprobar el pasaporte o algún documento oficial semejante, y de otra parte firmar claves sólo de individuos a los que se haya conocido personalmente durante un largo periodo de tiempo. La mayoría de miembros de Debian se encuentra en una posición intermedia entre estos dos extremos, y es una cuestión sobre la que no existe una posición oficial del Proyecto, sino que depende de los requerimientos de cada uno. Así, incluso tras intercambiar claves en una Fiesta de Firmado por ejemplo, no se sabe si la firma se va a producir realmente hasta que se envía.

En una entrevista encontramos una explicación muy sugerente de estas dos políticas básicas respecto al firmado de claves:

> **jonas:** It's about different opinions about what does it mean to sign a key. Some consider the keysigning to be linking the virtual world with the physical world. [...] Some consider keysigning as linking the virtual world, cyberspace, with government space. And others see it as linking cyberspace with community space, I think that is the key. So if you see it as, we are linking cyberspace with the government space, then it is stupid to say, we need to wait and when we meet again the third time I'll sign your key, or, we need to have dinner together before I sign your key.
>
> But if you are tying the cyberspace with the community space, you are trying to make the keys reflect how strong is this community. Then what you want is some way of making sure that it is your community, and nobody is faking the community. [...] If we don't meet again in three years, then there will be no keysigning, because we are not part of each other's community.

La entrevista continua discutiendo en contra esta segunda perspectiva, y de

hecho en la gran mayoría de las entrevistas realizadas se insistía en que esta operación de identificación era sólo eso, y no debía en ningún caso ser confundida con la confianza en las buenas intenciones o prácticas de los individuos.

Esas dos posiciones básicas tienen implicaciones diferentes respecto a cuestiones como la posibilidad de firmar la clave de alguien que sólo es conocido, en el Proyecto, mediante un pseudónimo, o no se conoce su nombre oficial. O respecto a la posibilidad de firmar una clave sin que exista un encuentro cara a cara.[48] O qué tipo de documentos oficiales se reconocen como válidos. Podemos observar aquí también que la relación entre identidad y confianza es compleja. Porque en tanto que la firma de una clave depende del juicio del que la firma respecto de la identidad del otro, se basa necesariamente en un juicio sobre la confianza, bien respecto de un documento, bien respecto de la confianza más personal inspirada por la trayectoria de quien posee la clave. Dentro de Debian, es cada vez más frecuente el requerimiento de un conocimiento previo, y una cierta confianza preexistente (véase por ejemplo Wolf (2014)). Se trata de poder reconocer a alguien a lo largo del tiempo, y de saber algo más de él de lo que suponen los documentos oficiales.

Además, incluso desde la perspectiva de que sólo es importante asegurarse de la identidad oficial de una persona mediante un documento burocrático hay implícita una presuposición de confianza. Al ser esta identificación transitiva, sería necesario confiar al menos en que la persona a la que se firma la clave sabe cómo usarla y protegerla,[49] y sobre todo que tiene una política de firmado de otras claves razonable, puesto que, debido a la transitividad de la confianza en el Anillo de Confianza, firmar su clave implica que se aceptarán sus firmas a otras claves como comprobación (parcial, son necesarias firmas de varias personas) de la identidad de terceras personas:

> **rmayorga:** Comprobar la identidad nada más, con un documento. Es lo único necesario, porque lo que se está verificando es que fulano es quien dice ser, pero no estoy verificando si fulano es confiable o es buena persona. Simplemente que es quien dice ser. [...] Yo muchas veces

[48] Es interesante respecto a éste y otros aspectos la discusión que sigue a Debian-devel (2016e).
[49] Sobre esto véase Debian-devel (2016f).

> no le firmo a alguien si veo que la persona tiene una llave generada hace como dos días. Primero le pregunto cosas como, para qué vas a usar la llave. Preguntas básicas sobre la seguridad de su llave. [...] Tienes que tener una cierta confianza en que sabe cómo usarla y cuidarla.

> **sto**: La confianza es transitiva. Si tú conoces a alguien que yo no conozco y yo me fío de ti, pues yo puedo transitivamente confiar en la otra persona. Si yo no me fío, que tú le firmes a un tercero la identidad, no me sirve de nada. [...] El problema es que también se usa para la otra, para reforzar el anillo y hacer que la confianza sea transitiva. Claro, si tú le firmas sólo a la gente que conoces, y ellos hacen lo mismo, el anillo es más fuerte en el sentido de que yo sé que toda la gente a la que yo le he firmado es de confianza, y por tanto si ellos firman yo sé que puedo confiar en que quien dicen ellos que es de confianza, bueno, no es de confianza, es quien dice ser. [...] Es un mecanismo que no se basa en una autoridad central, sino en la confianza uno a uno que es lo que debe tener la gente cuando está trabajando dentro del Proyecto.

El hecho es que se ha producido, a lo largo del tiempo, un desplazamiento. Si antes firmar una clave era sobre todo un asunto de comprobación de documentos y la *web of trust* se basaba sobre todo en ello, muchos miembros de Debian han ido cambiando de criterio. Mientras antes era más probable que les interesara sólo la validez del documento, cada vez es más común el interés por poder reconocer a la persona a lo largo del tiempo, por haber tratado con ella. En principio, esto supone solamente una manera diferente de asegurar que esa persona es quien dice ser, sin ninguna presuposición sobre su manera de actuar. Pero esto a su vez implica que se tiene la posibilidad de conocer algo más sobre ella, y con mucha seguridad que se relaciona habitualmente con el Proyecto. El punto de inflexión, al que se alude en una gran cantidad de entrevistas realizadas, estaría precisamente en el experimento realizado por Martin Krafft. Posiblemente lo que se encuentre aquí sea en parte una racionalización *a posteriori* de lo ocurrido. Algunos miembros contaban que habían firmado esa clave precisamente porque ese Desarrollador era muy conocido y lo reconocían por haberse encontrado en ocasiones previas.

Muchos miembros de Debian insisten en que se trata sólo de confianza en un documento legal, no en la persona. En una afirmación muy sintética:

> **zack:** That is a kind of misconception, because trust in GPG is not the trust in the person. Is the, how much you trust that the name on the key matches the person which claims that name.

Pero a pesar de la insistencia en separar identidad de confianza, el significado del término confianza (y su vínculo con las relaciones personales cara a cara) en el Anillo de Confianza ha ido influyendo en estas concepciones. En el siguiente fragmento de entrevista se puede ver que el sentido primario de confianza desborda el sentido más restringido en el que se insiste:

> **mooch:** Yo pretendía averiguar si gente que ha trabajado conmigo todos estos años aceptaría un documento fácilmente falsificable o incluso sin documento, para establecer una relación de confianza. Para que yo pueda seguir trabajando en el proyecto sin necesidad de presentar un documento oficial. [Algunos] me decían, no necesito carnet. La mayoría me dijo, sé quién eres, y entonces establecieron una relación de confianza conmigo basada simplemente en mi pasado, en reconocer que yo era la persona que había estado trabajando con ellos durante varios años, que yo era quien decía ser, simplemente por el conocimiento que ellos tenían de mi trabajo, de mi relación con ellos anteriormente. [...]
>
> Hemos tenido una discusión sobre qué es más importante para un proyecto como Debian, si establecer una relación de confianza basada en un documento físico que puedes falsificar, [...] o si esa relación ha sido fundamentalmente basada en una confianza que ha nacido y ha crecido en relación al trabajo que he hecho con ellos.
>
> Yo creo que eso se puede enlazar, o sea la persona y el comportamiento que va a tener esa persona se puede enlazar de una forma mucho más correcta basándote en cómo ha trabajado esa persona anteriormente, que por un documento legal. Si yo tengo una comunidad con la que

> colaboro y todo el mundo me conoce, la posibilidad de que yo haga algo en contra de unos valores comunes con esa comunidad es mucho más difícil, que si yo no conozco a esa comunidad y lo único que me relaciona con esa comunidad es un documento legal.

El vínculo entre estas dos formas de confianza estaría precisamente en que ambas, tanto el reconocimiento de la identidad de una persona, como la confianza en su manera de actuar, se fundamentan en la existencia de una relación personal más o menos continuada, tanto con uno mismo como con el Proyecto en su conjunto.

7 A modo de conclusión. El don del *software* libre y la mediación técnica de la colaboración.

> *El* mundo de las Cárites*, sin embargo, muestra todo su ser sólo si comprendemos que la «gracia», como Forma divina, no significa únicamente lo gracioso-encantador, lo que hace feliz con sus dones, sino también la alegría y la gratitud de sentirse feliz y obsequiado. Es el reino maravilloso del regalar y agradecer en uno, dar con amor y recibir con amor; el reino vedado al derecho y la justicia, a la pretensión y el desquite, el reino de la gracia plena.*
>
> —Otto (2007, pág. 90).

En la introducción señalábamos dos grandes objetivos de este trabajo. El primero era establecer algunas de las dimensiones éticas y políticas de las prácticas y acciones sociales que podemos observar en un colectivo sociotécnico constituido en torno a la producción y distribución de *software* libre. Esto ha supuesto vincularlas con conceptos como los de don o procomún. El segundo consistía en reconstruir el papel que en esas prácticas y acciones sociales tienen los dispositivos tecnológicos a través de los que se produce la mediación técnica. Ha quedado suficientemente mostrado que ambas cuestiones están completamente entrelazadas. Esta inseparabilidad se observa en primer lugar en el hecho de que no se pueden tratar aisladamente en capítulos diferentes, apareciendo su vinculación

en cualquiera de ellos y a propósito de cualquiera de los procesos descritos etnográficamente.

Intentaremos ahora hacer algunas consideraciones más generales a modo de conclusión. Empezaremos insistiendo en una forma de articulación entre esos dos objetivos mencionados previamente, relativa al papel de *hackers* y desarrolladores de *software* libre. Lo que hemos encontrado en sus prácticas es una «ingeniería heterogénea» que les permite «to blend together major social questions concerning the spirit of the age or the century and "properly" technological questions in a single discourse» (Latour, 1996, pág. 33). Esa mezcla, esa articulación de la que hablábamos, se produce entonces en esas prácticas sociales específicas. A lo largo de todo este trabajo lo que hemos intentado es precisamente mostrar y describir en detalle las diversas formas de mediación y cadenas de prácticas que lo permiten.

La pregunta que nos podemos hacer ahora es, ¿en qué puntos, en qué prácticas sociales más amplias pueden tener un impacto las prácticas que hemos descrito? ¿Qué suponen más allá de su campo de producción y aplicación, es decir, de la creación de *software* y otros dispositivos tecnológicos?

Creo que las dos cuestiones principales que se plantean serían la doocracia como forma de organización, y el papel del procomún en el mundo contemporáneo. En realidad, ambas cuestiones son interdependientes. Que el procomún (pensamos en el procomún digital, pero posiblemente esto sería aplicable también a otras formas de procomún) llegue a consolidarse efectivamente como constituyente de diferentes formas de acción social depende en buena medida de encontrar formas adecuadas de mediación técnica de la colaboración (como las que hemos visto asociadas a la doocracia). Que la doocracia pueda ser una forma extensible a otras prácticas depende de que se articule con un procomún. El don del *software* libre nos remite así inevitablemente a las formas de mediación técnica de la colaboración.

Considerada de forma abstracta (es decir, separada de su implementación concreta), la doocracia como forma de organizar vínculos sociales no es en sí misma especialmente interesante ni novedosa, y podría considerarse perfectamente

como otro nombre de la meritocracia en el mejor de los casos, o como una justificación de determinadas formas de poder y de concentración del mismo en el peor. Si la doocracia tiene un sentido positivo que va más allá de una forma de internalización (más o menos culpabilizadora) de la responsabilidad, es en tanto que pueda ser entendida y construida como una de las formas en las que se encarna la democracia en sentido ontológico. Según vimos, como la afirmación de la igualdad esencial de los agentes y la negación de posiciones de privilegio o acumulación de poder previas. Supone así el reconocimiento de que cualquiera puede ocupar esas posiciones. Es a partir de ahí que puede aparecer la potencialidad utópica del concepto, ligada al imaginario social en torno a la falta de jerarquía que forma parte de estas formas de vida. La doocracia se vincula con formas de constituir actores que los convierte siempre en mediadores, no en meros intermediarios de los deseos u objetivos de otros. Supone así el poder y la agencia para hacer, para actuar, pero no para mandar. Es entonces menos capitalizable individualmente que la meritocracia, superando muchos de los problemas de ésta.

Así, el punto fundamental de esta forma de organización y decisión se encuentra en su capacidad de iluminar y superar, transformándolas, algunas de las tensiones entre saber técnico y democracia, que al menos desde Sócrates y Platón han condicionado el pensamiento político (y que como hemos visto se encuentran también presentes en el colectivo constituido en torno al Proyecto Debian). En la medida en que la agencia y la capacidad de decisión se articulan necesariamente con saberes y prácticas técnicas, éstos están efectivamente abiertos a todos los agentes. De ahí la insistencia constante en la apertura, y la relevancia del concepto de público recursivo. Como hemos visto, la doocracia puede contribuir a una forma de organización no jerárquica si los dispositivos tecnológicos de los que depende son abiertos. La doocracia supone que se han de tener los suficientes derechos de acceso a esos dispositivos. Así, doocracia y público recursivo son dos aspectos inseparables de los mismos procesos, y ambos se apoyan en la creatividad, en la capacidad de crear normas, convenciones y dispositivos.

En cuanto al papel que juegan los dispositivos tecnológicos, éste es quizás uno de los puntos más importantes que el fenómeno del *software* libre ha puesto de manifiesto, el hecho de que la ética y la política se realizan también construyendo dispositivos. Ésta es también una idea presente en muchos de los autores que

hemos citado y utilizado en este trabajo. Las instituciones y las prácticas políticas y éticas no se construyen sólo a través de medios éticos y políticos en sí mismos, sino que necesitan su encarnación en toda una serie de dispositivos mediadores. Que existan jerarquía, libertad, poder o igualdad, sus formas y grados, depende en buena parte del tipo de no humanos (materiales y no materiales) que se incluyan en el colectivo. Como vimos, una manera de caracterizar las prácticas en el Proyecto Debian es como un proceso de construcción de formas de mediación técnica y legal que traduzcan, reelaboren y definan toda una serie de fines éticos y políticos. De ahí que términos como los de parche o doocracia puedan ser útiles al traducirlos y trasladarlos a otros ámbitos de acción. La atención dedicada en este trabajo a los dispositivos tecnológicos no es algo accesorio, sino que es fundamental para entender la acción social y la constitución del colectivo sociotécnico. Sólo a partir del detalle de esa descripción podemos entender la doocracia y su relación con la democracia.

En términos generales, y en relación a la teoría semiótica de la agencia, la doocracia es también entonces, como decíamos, un sistema de organización que pretende dar cuenta del hecho de que el valor y el significado se crean a partir de la acción y de la capacidad de establecer nuevas conexiones y asociaciones entre elementos heterogéneos. Es decir, no de la acción individual, sino colectiva. Recordemos que el lugar propio donde se encuentra la agencia es el colectivo. La creatividad, como la agencia, es distribuida (Leach, 2005). Por otra parte, una teoría semiótica de la agencia es también, necesariamente, una teoría ontológica. Todo proceso semiótico es simultáneamente un proceso de creación de objetos (según vimos, a partir del compromiso con determinados interpretantes).

Pero si la agencia y la creatividad son distribuidas y colectivas, entonces la doocracia sólo puede encarnar este sentido de democracia si además se articula con un ámbito de procomún. Es decir, según lo que hemos visto, con la existencia de un modo u otro de la obligación de recibir. Si la libertad supone la posibilidad de establecer vínculos, la posibilidad de negar la obligación de recibir supone el rechazo de la libertad de los demás para establecer por sí mismos esos vínculos. La doocracia y el procomún permiten entonces dar cuerpo a un concepto de libertad que no está limitado a la concepción liberal de la propiedad de uno mismo y la ausencia de interferencias. Esta concepción de la libertad se incorpora material-

mente en toda una serie de dispositivos y agenciamientos. Así, se puede considerar que hemos venido describiendo agenciamientos sociotécnicos que potencian la libertad, la solidaridad y la cooperación. Un colectivo estabilizado y constituido en torno al procomún, el don, la libertad y la doocracia, es decir la democracia. Por otra parte, si el papel estructural central del procomún nos condujo a la cuestión de la gracia, podemos señalar aquí que la *kháris* es también la alegría, en estrecha relación con la creatividad.[1]

Ya hemos considerado detalladamente cómo la dependencia del don del *software* libre respecto de la existencia de un ámbito de procomún establece toda una serie de condicionamientos específicos en su funcionamiento como objeto del don. Es un tipo de don que, precisamente por su vinculación con el procomún y aunque supone las tres obligaciones que distinguía Mauss respecto al don, no crea sin embargo deuda personal, en tanto que, como vimos, las relaciones establecidas a través de este don están siempre mediadas por el procomún.[2] Ésta es una de la razones de que la doocracia, que en nuestro campo hemos visto ligada también al don, supone el poder y la agencia para hacer, pero no para mandar o coaccionar. La doocracia en su relación con el don es menos susceptible de favorecer la apropiación privada que la meritocracia y otras formas de reparto de agencia y poder. Precisamente por esto, la existencia de un procomún en este circuito de los dones es una manera de resolver las tensiones y ambigüedades del fenómeno del don en general, especialmente las ligadas a la producción de jerarquías y desigualdad. El contenido utópico del *software* libre y de alguna de las versiones de la ética *hacker* (que podemos entender como una ética de la colaboración, la autonomía y la implicación intensa con determinadas libertades, fundamentalmente en torno al conocimiento y la cultura libres), no se encuentra entonces en ni depende de la pura generosidad o el altruismo, sino que se manifiesta en la creación y el desarrollo de agenciamientos que son también materiales. De ahí también que, respecto a la cuestión del impacto de la aparición y la consolidación del fenómeno del *software* libre, éste se ha mostrado como uno de los modelos (el de más éxito) y al mismo tiempo como infraestructura del campo de la cultura libre.

[1] Sobre la relación de estos elementos, véase Otto (2007, págs.87-90).

[2] Sobre la relación entre deuda y don en el mundo contemporáneo, véase High (2012). Para la relación entre el don, la deuda y el crédito por un lado, y la producción criptográfica de la confianza por el otro, véase Kelty (2005b).

7. A modo de conclusión. El don del *software* libre y la mediación técnica de la colaboración.

En definitiva, el procomún, la cultura libre, el don y la doocracia no sólo son una manera de entender alguna de las controversias sobre el régimen de propiedad intelectual propio del mundo contemporáneo, sino que también constituyen un ámbito privilegiado donde observar y descubrir nuevas formas de relación y mediación entre los actores implicados, y la importancia de entender la constitución de nuevas redes sociotécnicas haciendo justicia tanto a las formas de acción de los sujetos como a la realidad de sus componentes tecnológicos.

Retomando lo que decíamos al iniciar estas conclusiones, podemos recordar lo que veíamos a propósito de la habilidad de los miembros de Debian en el ámbito de las relaciones sociales: aunque ellos se entienden en numerosas ocasiones como sujetos técnicos poco habilidosos respecto a las relaciones sociales, podemos interpretar también su acción como una manera muy eficaz de crear y estabilizar las relaciones que conforman el colectivo sociotécnico. Lo que se puede observar a lo largo de toda esta investigación es que en todo caso lo «técnico» contribuye a explicar lo «social», mucho más que a la inversa entendiendo lo «social» como contexto. En este sentido, se puede considerar, respecto a los resultados de esta investigación, que se trata mucho más de aprender sobre la constitución de lo social a partir de las prácticas estudiadas, que de interpretar o explicar éstas a partir de la teoría social o antropológica previa.

Bibliografía

A no ser que se especifique lo contrario, todas las referencias *web* han sido comprobadas en junio de 2018.

Akrich, Madeleine (1992). «The De-Scription of Technical Objects». En *Shaping Technology / Building Society*. Editado por Wiebe E. Bijker y John Law. Cambridge, Mass.: The MIT Press, págs. 205-224.

Akrich, Madeleine y Bruno Latour (1992). «A Summary of a Convenient Vocabulary for the Semiotics of Human and Nonhuman Assemblies». En *Shaping Technology / Building Society*. Editado por Wiebe E. Bijker y John Law. Cambridge, Mass.: The MIT Press, págs. 259-264.

Amit, Vered (2002). «Reconceptualizing community». En *Realizing Community: Concepts, Social Relationships and Sentiments*. Editado por Vered Amit. New York: Routledge.

Ardèvol, E., M. Bertrán, M. Callén y C. Pérez (2003). «Etnografía virtualizada: la investigación etnográfica de internet». *Athenea Digital* 3, págs. 72-92.

Banks, Marcus (1996). *Ethnicity: Anthropological Constructions*. London: Routledge.

Barth, Fredrik, ed. (1976). *Los grupos étnicos y sus fronteras: La organización social de las diferencias culturales*. México: Fondo de Cultura Económica.

Baym, Nancy K. (2010). *Personal Connections in the Digital Age*. Cambridge: Polity Press.

Benkler, Yochai (2006). *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. New Haven: Yale University Press.

Berdou, Evangelia (2011). *Organization in Open Source Communities. At the Crossroads of the Gift and Market Economies*. New York: Routledge.

Bergquist, M. y J. Ljungberg (2001). «The Power of Gifts: Organizing Social Relationships in Open Source Communities». *Information Systems Journal* 11(4), págs. 305-320.

Bijker, Wiebe E., Thomas P. Hughes y Trevor J. Pinch, eds. (1989). *The Social Construction of Technological Systems*. Cambridge, Mass.: The MIT Press.

Bijker, Wiebe E. y John Law, eds. (1992). *Shaping Technology / Building Society*. Cambridge, Mass.: The MIT Press.

Bourdieu, Pierre (1977). *Outline of a Theory of Practice*. Cambridge: Cambridge University Press.

Boyle, James (2005). «Fencing Off Ideas: Enclosure and the Disappearance of the Public Domain». En *CODE: Collaborative Ownership and the Digital Economy*. Editado por Rishab Aiyer Ghosh. Cambridge, Mass.: The MIT Press.

— (2008). *The Public Domain. Enclosing the Commons of the Mind*. New Haven y London: Yale University Press.

Brown, John Seely y Paul Duguid (2001). «Knowledge and Organization: A Social-Practice Perspective». *Organization Science* 12(2), págs. 198-213.

Brubaker, R. y F. Cooper (2000). «Beyond "Identity"». *Theory and Society* 29(1), págs. 1-47.

Callon, Michel (1984). «Some elements of a sociology of translation: domestication of the scallops and the fishermen of St Brieuc Bay». *The Sociological Review* 32.

— (2005). «Why Virtual paves the way to Political Impotence: A Reply to Daniel Miller's Critique of *The Laws of the Markets*». *Economic Sociology: European Electronic Newsletter* 6(2), págs. 3-20.

Callon, Michel y Bruno Latour (1981). «Unscrewing the big Leviathan: How actors macrostructure reality and sociologists help them to do so». En *Advances in social theory and methodology: Towards an integration of micro- and macro-sociologies*. Editado por Karin Knorr-Cetina y Aron V. Cicourel. London: Routledge & Kegan Paul.

Casajus, Dominique (2005). «Sagrado». En *Diccionario Akal de Etnología y Antropología*. Editado por Pierre Bonte y Michael Izard. Madrid: Akal.

Clemmitt, Marcia (2011). «Computer Hacking». *CQ Researcher* 21(32), págs. 757-780.

Clifford, James (1995). *Dilemas de la cultura. Antropología, literatura y arte en la perspectiva posmoderna*. Barcelona: Gedisa.

Clifford, James y George E. Marcus, eds. (1986). *Writing Culture*. Berkeley: University of California Press.

Clippinger, John y David Bollier (2005). «A Renaissance of the Commons: How the New Sciences and Internet are Framing a New Global Identity and Order». En *CODE: Collaborative Ownership and the Digital Economy*. Editado por Rishab Aiyer Ghosh. Cambridge, Mass.: The MIT Press.

Cohen, Anthony P. (1985). *The Symbolic Construction of Community*. London: Routledge.

Coleman, Gabriella (2003). «The (copylefted) Source Code for the Ethical Production of Information Freedom». En *Shaping Technologies*. Editado por Ravi Vasudevan y otros. Sarai Reader – 03. Dehli: The Sarai Programme, Centre for the Study of Developing Societies.

— (2005). «The Social Construction of Freedom in Free and Open Source Software: Hackers, Ethics, and the liberal Tradition». Department of Anthropology, Chicago University.

— (2009). «Code is speech: legal tinkering, expertise, and protest among free and open source software developers». *Cultural Anthropology* 24(3), págs. 420-54.

— (2010a). «Ethnographic Approaches to Digital Media». *Annual Review of Anthropology* 39, págs. 487-505.

— (2010b). «The Hacker Conference: A Ritual Condensation and Celebration of a Lifeworld». *Anthropological Quarterly* 83(1), págs. 99-124.

— (2013). *Coding Freedom. The Ethics and Aesthetics of Hacking*. Princeton: Princeton University Press.

Coleman, Gabriella y Alex Golub (2008). «Hacker practice: Moral genres and the cultural articulation of liberalism». *Anthropological Theory* 8(3).

Coleman, Gabriella y Benjamin Mako Hill (2005). «The Social Production Of Ethics In Debian And Free Software Communities: Anthropological Lessons For Vocational Ethics». En *Free/Open Source Software Development*. Editado por Stephen Koch. London: Idea Group Publishing.

Collins-Sussman, Ben, Brian W. Fitzpatrick y C. Michael Pilato (2008). *Version Control with Subversion*. O'Reilly Media.

Cruces, Francisco (2003). «Etnografías sin final feliz. Sobre las condiciones de posibilidad del trabajo de campo urbano en contextos globalizados». *Revista de Dialectología y Tradiciones Populares* LVIII(2), págs. 161-178.

Czarniawska, B. y T. Hernes, eds. (2005). *Actor-Network Theory and Organizing*. Malmo, Den.: Liber & Copenhagen Business School Press.

De Paoli, Stefano y Vincenzo D'Andrea (2008). «How artefacts rule web-based communities: practices of free software development». *International Journal of Web Based Communities* 4(2), págs. 199-219.

De Paoli, Stefano, Maurizio Teli y Vincenzo D'Andrea (2008). «Free and open source licenses in community life: Two empirical cases». *First Monday* 13(10). url: http://journals.uic.edu/ojs/index.php/fm/article/view/2064.

DeMillo, Richard A., Richard J. Lipton y Alan J. Perlis (1977). «Social Processes and Proofs of Theorems and Programs». En *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. POPL '77. Los Angeles, California: ACM, págs. 206-214.

DeNardis, Laura (2009). *Protocol Politics*. Cambridge, Mass.: The MIT Press.

Derrida, Jacques (1995). *Dar (el) tiempo. I. La falsa moneda*. Barcelona: Paidós.

Dexter, Scott, Melissa Dolese, Angelika Seidel y Aaron Kozbelt (2011). «On the Embodied Aesthetics of Code». *Culture Machine* 12.

Díaz de Rada, Ángel (2007). «Valer y valor. Una exhumación de la teoría del valor para reflexionar sobre la desigualdad y la diferencia en relación con la escuela». *Revista de Antropología Social* 16, págs. 117-158.

— (2010). *Cultura, antropología y otras tonterías*. Madrid: Trotta.

— (2013). «Acción social, cultura escolar y documento: semiosis y etnografía en el examen de los espacios documentales». En *La historia de la cultura escolar en España y en Italia. Balances y Perspectivas*. Editado por Juri Meda y Ana María Badanelli. Macerata: EUM Edizioni Università di Macerata, págs. 229-255.

DiBona, Chris, Sam Ockman y Mark Stone, eds. (1999). *Open Sources. Voices from the Open Source Revolution*. O'Reilly.

Edwards, Kasper (2001). «Epistemic Communities, Situated Learning and Open Source Software Development». *Proceedings from the conference on Epistemic Cultures and the Practice of Interdisciplinarity*.

Ess, Charles y May Thorseth, eds. (2011). *Trust and Virtual Worlds*. New York: Peter Lang.

Fischer, Michael M. J. (2007). «Four genealogies for a recombinant anthropology of science and technology». *Cultural Anthropology* 22, págs. 539-615.

Fuller, Matthew (2003). *Behind the Blip. Essays on the Culture of Software*. New York: Autonomedia.
— (2008a). «Introduction». En *Software Studies. A Lexicon*. Editado por Matthew Fuller. Cambridge, Mass.: The MIT Press.
— ed. (2008b). *Software Studies. A Lexicon*. Cambridge, Mass.: The MIT Press.
Geertz, Clifford (1994). *Conocimiento local*. Barcelona: Paidós.
Ghosh, Rishab Aiyer, ed. (2005a). *CODE: Collaborative Ownership and the Digital Economy*. Cambridge, Mass.: The MIT Press.
— (2005b). «Why collaboration is important (again)». En *CODE: Collaborative Ownership and the Digital Economy*. Editado por Rishab Aiyer Ghosh. Cambridge, Mass.: The MIT Press.
Godelier, Maurice (1998). *El enigma del don*. Barcelona: Paidós.
González de Requena, Fernando (2012). «Política digital y nuevas prácticas tecnológicas». En *Jóvenes, Culturas Urbanas y Redes Digitales*. Editado por Néstor García Canclini, Francisco Cruces y Maritza Urteaga. Madrid: Ariel.
— (2016). «Software, hackivismo y políticas digitales». En *Cosmópolis. Nuevas maneras de ser urbanos*. Editado por Francisco Cruces. Barcelona: Gedisa.
Gonzalez-Barahona, J., G. Robles, M. Michlmayr, J. J. Amor y D. German (2009). «Macro-level software evolution: a case study of a large software compilation». *Empirical Software Engineering* 14(3), págs. 262-285.
Graeber, David (2001). *Toward an Anthropological Theory of Value. The False Coin of Our Own Dreams*. New York: Palgrave.
— (2011). *Debt: The First 5,000 years*. New York: Melville House.
— (2013). «It is value that brings universes into being». *HAU: Journal of Ethnographic Theory* 3(2), págs. 219-43.
Gregory, Christopher (1982). *Gifts and Commodities*. Londres y Nueva York: Academic Press.
— (1997). *Savage money: the anthropology and politics of commodity exchange*. Amsterdam: Harwood Academic.
Haas, Peter M. (1992). «Introduction: epistemic communities and international policy coordination». *International Organization* 46(1), págs. 1-35.
Hacking, Ian (1999). *The social construction of what?* Cambridge, MA: Harvard University Press.
Hardin, Garrett (1968). «The tragedy of the commons». *Science* (162), págs. 1243-1248.
Hertzog, Raphaël y Roland Mas (2012). *The Debian Administrator's Handbook*. Freexian SARL.

High, Holly (2012). «Re-reading the potlatch in a time of crisis: debt and the distinctions that matter1». *Social Anthropology* 20(4), págs. 363-379.

Himanen, Pekka (2003). *La ética del hacker y el espíritu de la era de la información*. Barcelona: Destino.

Hine, Christine (2004). *Etnografía virtual*. Barcelona: UOC.

Holmes, David, ed. (1997). *Virtual Politics. Identity and Community in Cyberspace*. London: Sage.

Høstaker, Roar (2005). «Latour - Semiotics And Science Studies». *Science Studies* 18(2), págs. 5-25.

Hughes, Thomas P. (1989). «The Evolution of Large Technological Systems». En *The Social Construction of Technological Systems*. Editado por Wiebe E. Bijker, Thomas P. Hughes y Trevor J. Pinch. Cambridge, Mass.: The MIT Press.

IEEE Standards Information Network/IEEE Press (2000). *The Authoritative Dictionary of IEEE Standards Terms (IEEE 100)*. Institute of Electrical and Electronics Engineers (IEEE).

Jones, Steven. G., ed. (1995). *Cybersociety: Computer Mediated Communication and Community*. Thousand Oaks: SAGE Publications.

— ed. (1998). *Cibersociety 2.0. Revisiting Computer-Mediated Communication and Community*. Thousand Oaks: SAGE Publications.

Kahn Gillmor, Daniel (2007). «Technical Architecture Shapes Social Structure». En López, Alfredo, Jamie McClelland, Eric Goldhagen, Daniel Kahn Gillmor y Amanda B. Hickman. *The Organic Internet*. Entremundos Publications. url: http://lair.fifthhorseman.net/~dkg/tls-centralization/.

Kelty, Christopher (2002). «Hau to do things with Words». url: http://kelty.org/or/papers/unpublishable/Kelty.Hautodothings.2002.pdf.

— (2005a). «Geeks, Social Imaginaries, and Recursive Publics». *Cultural Anthropology* 20(2), págs. 185-214.

— (2005b). «Trust among the Algorithms: Ownership, Identity, and the Collaborative Stewardship of Information». En *CODE: Collaborative Ownership and the Digital Economy*. Editado por Rishab Aiyer Ghosh. Cambridge, Mass.: The MIT Press.

— (2006). «The Scale of Norms: Free Software and the theories of Gift Exchange». url: http://kelty.org/or/papers/unpublishable/Kelty-Gifts-Dec-2006-Revised.pdf.

— (2008). *Two Bits: The Cultural Significance of Free Software*. Durham: Duke University Press.

Kernighan, Brian W. y Rob Pike (1984). *The UNIX Programming Environment*. Prentice Hall, Inc.

Knorr-Cetina, Karin y Aron V. Cicourel, eds. (1981). *Advances in social theory and methodology: Towards an integration of micro- and macro-sociologies*. London: Routledge & Kegan Paul.

Knuth, Donald E. (1984). «Literate Programming». *The Computer Journal* 27(2), págs. 97-111.

Koch, S., ed. (2005). *Free/Open Source Software Development*. London: Idea Group Publishing.

Kockelman, Paul (2007). «Agency. The Relation between Meaning, Power, and Knowledge». *Current Anthropology* 48(3), págs. 375-401.

Kollock, Peter (1999). «The economies of online cooperation: gifts and public goods in cyberspace». En *Communities in Cyberspace*. Editado por Marc A. Smith y Peter Kollock. London: Routledge, págs. 220-239.

Krafft, Martin F. (2005). *The Debian System*. San Francisco: No Starch Press.

Krysa, Joasia y Grzesiek Sedek (2008). «Source Code». En *Software Studies. A Lexicon*. Editado por Matthew Fuller. Cambridge, Mass.: The MIT Press.

Kuhn, Bradley M., ed. (2015). *Copyleft and the GNU General Public License: A Comprehensive Tutorial and Guide*. url: https://copyleft.org/guide/.

Lafuente, Antonio (2007). «Los cuatro entornos del procomún». *Archipiélago* (77-78), págs. 15-22.

Lanzara, G.F. y M. Morner (2005). «Artifacts Rule! How Organizing Happens in Open Source Software Projects». En *Actor-Network Theory and Organizing*. Editado por B. Czarniawska y T. Hernes. Malmo, Den.: Liber & Copenhagen Business School Press, págs. 67-90.

Latour, Bruno (1988). *The Pasteurization of France*. Cambridge, Mass.: Harvard University Press.

— (1992). «Where Are the Missing Masses? The Sociology of a Few Mundane Artifacts». En *Shaping Technology / Building Society*. Editado por Wiebe E. Bijker y John Law. Cambridge, Mass.: The MIT Press, págs. 225-258.

— (1996). *Aramis or the love of technology*. Cambridge, MA: Harvard University Press.

— (2001). *La esperanza de Pandora. Ensayo sobre la realidad de los estudios de la ciencia*. Barcelona: Gedisa.

— (2005). *Reassembling the Social*. Oxford: Oxford University Press.

Latour, Bruno (2007). *Nunca fuimos modernos. Ensayo de antropología simétrica*. Buenos Aires: Siglo XXI.

Lave, J. y E. Wenger (1991). *Situated Learning: Legitimate Peripheral Participation*. Cambridge: Cambridge University Press.

Law, John, ed. (1986). *Power, Action and Belief: A New Sociology of Knowledge*. London: Routledge & Kegan Paul.

— (2008). «Actor-network theory and material semiotics». En *The New Blackwell Companion to Social Theory*. Editado por Bryan S. Turner. Oxford: Blackwell, págs. 141-158.

Leach, James (2005). «Modes of creativity and the register of ownership». En *CODE: Collaborative Ownership and the Digital Economy*. Editado por Rishab Aiyer Ghosh. Cambridge, Mass.: The MIT Press.

Leach, James, Dawn Nafus y Bernhard Krieger (2009). «Freedom Imagined: Morality and Aesthetics in Open Source Software Design». *Ethnos* 74(I), págs. 51-71.

Lessig, Lawrence (2005). *Por una cultura libre. Cómo los grandes grupos de comunicación utilizan la tecnología y la ley para clausurar la cultura y controlar la creatividad*. Madrid: Traficantes de sueños.

— (2009). *El código 2.0*. Madrid: Traficantes de Sueños.

Lévi-Strauss, Claude (1971). «Introducción a la obra de Marcel Mauss». En Mauss, Marcel. *Sociología y Antropología*. Madrid: Tecnos, págs. 13-42.

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

López, Alfredo, Jamie McClelland, Eric Goldhagen, Daniel Kahn Gillmor y Amanda B. Hickman (2007). *The Organic Internet*. Entremundos Publications.

Mackenzie, Adrian (2006). *Cutting Code. Software and Sociality*. New York: Peter Lang.

MacKenzie, Donald (2004). *Mechanizing Proof. Computing, Risk, and Trust*. Cambridge, Mass.: The MIT Press.

— ed. (2009a). *Material Markets. How Economic Actors are Constructed*. Oxford: Oxford University Press.

— (2009b). «Ten Precepts for the Social Studies of Finance». En *Material Markets. How Economic Actors are Constructed*. Editado por Donald MacKenzie. Oxford: Oxford University Press.

MacKenzie, Donald y Iain Hardie (2009). «Assembling an Economic Actor». En *Material Markets. How Economic Actors are Constructed*. Editado por Donald MacKenzie. Oxford: Oxford University Press.

Manovich, Lev (2001). *The Language of New Media*. Cambridge, Mass.: The MIT Press.

— (2013). *Software takes Command*. New York: Bloomsbury.

Mansell, Robin y Evangelia Berdou (2010). «Political Economy, the Internet and FL/OSS Development». En *International Handbook of Internet Research*. Editado por Jeremy Hunsinger, Lisbeth Klastrup y Matthew M. Allen. New York: Springer, págs. 341-361.

Marcus, George E. (1995). «Ethnography in/of the World System: The Emergence of Multi-Sited Ethnography». *Annual Review of Anthropology* 24, págs. 95-117.

Marcus, George E. y Michael M. J. Fischer (1986). *Anthropology as Cultural Critique*. Chicago: The University of Chicago Press.

Marino, Mark C. (4 de dic. de 2006). «Critical Code Studies». *Electronic book review*. url: http://www.electronicbookreview.com/thread/electropoetics/codology.

Mateos-Garcia, Juan y W. Edward Steinmueller (2008). «The institutions of open source software: Examining the Debian community». *Information Economics and Policy* 20(4), págs. 333-344.

Mauss, Marcel (2009). *Ensayo sobre el don. Forma y función del intercambio en las sociedades arcaicas*. Buenos Aires: Katz.

Michlmayr, Martin y Benjamin Mako Hill (2003). «Quality and the reliance on individuals in free software projects». En *Proceedings of the 3rd Workshop on Open Source Software Engineering*, págs. 105-109.

Miller, Daniel y Don Slater (2000). *The Internet. An Ethnographic Approach*. Oxford: Berg.

Moglen, Eben (1999). «Anarchism triumphant: Free software and the death of copyright». *First Monday* 4(8). url: http://journals.uic.edu/ojs/index.php/fm/article/view/684.

Nussbaum, Lucas y Stefano Zacchiroli (2010). «The Ultimate Debian Database: Consolidating Bazaar Metadata for Quality Assurance and Data Mining». En *MSR 2010: 7th IEEE Working Conference on Mining Software Repositories*.

O'Mahony, Siobhán (2003). «Guarding the commons: how community managed software projects protect their work». *Research Policy* 32(7), págs. 1179-1198.

O'Mahony, Siobhán y Fabrizio Ferraro (2007). «The emergence of governance in an open source community». *The Academy of Management Journal* 50(5), págs. 1079-1106.

O'Neil, Mathieu (2009). *Cyberchiefs. Autonomy and Authority in Online Tribes*. London: Pluto Press.

Ong, Aihwa y Stephen J. Collier (2005a). «Global Assemblages, Anthropological Problems». En *Global Assemblages. Technology, Politics, and Ethics as Anthropological Problems*. Editado por Aihwa Ong y Stephen J. Collier. Oxford: Blackwell.

— eds. (2005b). *Global Assemblages. Technology, Politics, and Ethics as Anthropological Problems*. Oxford: Blackwell.

Ortega, Felipe y Joaquín Rodríguez (2011). *El Potlatch Digital. Wikipedia y el triunfo del procomún y el conocimiento compartido*. Madrid: Cátedra.

Ossewaarde, Marinus y Wessel Reijers (2017). «The illusion of the digital commons: 'False consciousness' in online alternative economies». *Organization* 24(5), págs. 609-628.

Otto, Walter F. (2007). *Teofanía. El espíritu de la antigua religión griega*. Madrid: Sexto Piso.

Pearsall, Judy, ed. (1998). *The New Oxford Dictionary of English*. Oxford: Oxford University Press.

Peristany, J. G. y J. Pitt-Rivers, eds. (1992). *Honor and Grace in Anthropology*. Cambridge: Cambridge University Press.

Pinch, Trevor J. y Wiebe E. Bijker (1989). «The Social Construction of Facts and Artifacts: Or How the Sociology of Science and the Sociology of Technology Might Benefit Each Other». En *The Social Construction of Technological Systems*. Editado por Wiebe E. Bijker, Thomas P. Hughes y Trevor J. Pinch. Cambridge, Mass.: The MIT Press.

Pitt-Rivers, J. (1992). «Postscript: the place of grace in anthropology». En *Honor and Grace in Anthropology*. Editado por J. G. Peristany y J. Pitt-Rivers. Cambridge: Cambridge University Press.

Ratto, Matt (2003). «Re-working by the Linux Kernel developers». url: http://flosshub.org/sites/flosshub.org/files/ratto.pdf.

— (2005a). «"Don't fear the penguins": Negotiating the trans-local space of linux development». *Current Anthropology* 46(5), págs. 827-834.

— (2005b). «Embedded technical expression: Code and the leveraging of functionality». *The Information Society* 21(3), págs. 205-213.

Raymond, Eric S. (1999). *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly Media.

— (2000). «Homesteading the Noosphere». url: http://catb.org/~esr/writings/homesteading/homesteading/index.html.
— (2003). *The Art of Unix Programming*. Addison-Wesley.
Raymond, Eric S. y Guy L. (comps.) Steele (2004). «The Jargon File, version 4.4.8, 01 Oct 2004». url: http://www.catb.org/jargon/ (visitado 06-08-2013).
Rheingold, Howard (2000). *The virtual community. Homesteading on the electronic frontier*. Cambridge, MA: The MIT Press.
Sadowski, Bert M., Gaby Sadowski-Rasters y Geert Duysters (2008). «Transition of governance in a mature open software source community: Evidence from the Debian case». *Information Economics and Policy* 20(4), págs. 323-332.
Sahlins, Marshall (1997). «The Spirit of the Gift». En *The logic of the gift*. Editado por Alan D. Schrift. New York y London: Routledge.
Schrift, Alan D., ed. (1997). *The logic of the gift*. New York y London: Routledge.
Shaik, M. y T. Cornford (2004). «Version control tools: a collaborative vehicle for learning in F/OS». En *Collaboration, Conflict and Control: Proceedings of the The 4th Workshop on Open Source Software Engineering*. Editado por J. Feller, B. Fitzgerald, S. Hissam y K. Lakhani. Edinburgh, Scotland, págs. 87-91.
Smith, Marc A. y Peter Kollock, eds. (1999). *Communities in Cyberspace*. London: Routledge.
Soria Guzmán, Irene, ed. (2016). *Ética hacker, seguridad y vigilancia*. México D.F.: Universidad del Claustro de Sor Juana.
Stalder, Felix (2010). «Digital Commons». En *The Human Economy*. Editado por Keith Hart, Jean-Louis Laville y Antonio David Cattani. Cambridge: Polity Press.
Star, Susan Leigh (1999). «The Ethnography of Infrastructure». *American Behavioral Scientist* 43(3), págs. 377-391.
Strathern, Marilyn (1990). *The Gender of the Gift*. Berkeley: University of California Press.
— (2004). *Partial Connections*. Walnut Creek: Altamira Press.
Sundara Rajan, Mira T. (2011). *Moral Rights: Principles, Practice and New Tehcnology*. Oxford: Oxford University Press.
Thomas, Douglas (2002). *Hacker Culture*. Minneapolis: University of Minnesota Press.
Torvalds, Linux (1999). «The Linux Edge». En *Open Sources. Voices from the Open Source Revolution*. Editado por Chris DiBona, Sam Ockman y Mark Stone. O'Reilly.

Turner, Bryan S., ed. (2008). *The New Blackwell Companion to Social Theory*. Oxford: Blackwell.

Turner, Fred (2005). «Actor-Networking the News». *Social Epistemology* 19(4), págs. 321-324.

— (2009). «Burning Man at Google: a cultural infrastructure for new media production». *New Media Society* 11, págs. 73-94.

Tyler, Stephen A. (1986). «Post-Modern Ethnography: From Document of the Occult to Occult Document». En *Writing Culture*. Editado por James Clifford y George E. Marcus. Berkeley: University of California Press.

Van den Brande, Ywein, Shane Coughlan y Till Jaeger, eds. (2014). *The International Free and Open Source Software Law Book*. Open Source Press. url: http://ifosslawbook.org/united-states-of-america/.

Velasco, Honorio y Ángel Díaz de Rada (2004). *La lógica de la investigación etnográfica*. Madrid: Trotta.

Velasco, Honorio, Ángel Díaz de Rada *et al.* (2006). *La sonrisa de la institución. Confianza y riesgo en sistemas expertos*. Madrid: Centro de estudios Ramón Areces.

Wallach, Hanna, Moray Allan y Dafydd Harries (2005). «The Debian New Maintainer Process: History and Aims». Presented at the 6th Annual International Debian Developers' Conference, 2005. url: http://dirichlet.net/pdf/wallach05debian.pdf.

Weber, Steven (2004). *The Success of Open Source*. Cambridge, MA: Harvard University Press.

Weiner, Annette B. (1992). *Inalienable Possessions. The Paradox of Keeping-While-Giving*. Berkeley: University of California Press.

Wilson, Samuel M. y Leighton C. Peterson (2002). «The Anthropology of Online Communities». *Annual Review of Anthropology* 31, págs. 449-67.

Winner, Langdon (1986). «Do Artifacts have politics?» En *The whale and the reactor: a search for limits in an age of high technology*. Chicago: University of Chicago Press, págs. 19-39.

Wittgenstein, Ludwig (2008). *Investigaciones filosóficas*. Barcelona: Crítica.

Wolf, Gunnar (2011b). «Factores de motivación y elementos de reconocimiento». En *Construcción colaborativa del conocimiento*. Editado por Gunnar Wolf y Alejandro Miranda. México: UNAM, Instituto de Investigaciones Económicas.

— (2016). «Cifrado e identidad, no todo es anonimato». En *Ética hacker, seguridad y vigilancia*. Editado por Irene Soria Guzmán. México D.F.: Universidad del Claustro de Sor Juana.

Wolf, Gunnar y Gina Gallegos-García (2017). «Strengthening a curated web of trust in a geographically distributed project». *Cryptologia* 41(5), págs. 459-475.

Wolf, Gunnar y Alejandro Miranda, eds. (2011). *Construcción colaborativa del conocimiento*. México: UNAM, Instituto de Investigaciones Económicas.

Wolf, Gunnar y Víctor González Quiroga (2018). «Insights on the large-scale deployment of a curated Web-of-Trust: the Debian project's cryptographic keyring». *Journal of Internet Services and Applications* 9(1), pág. 11.

Yuill, Simon (2008). «Concurrent Version System». En *Software Studies. A Lexicon*. Editado por Matthew Fuller. Cambridge, Mass.: The MIT Press.

Zeitlyn, David (2003). «Gift economies in the development of open source software: anthropological reflections». *Research Policy* 32(7), págs. 1287-1291.

Blog posts y artículos *online*

Adams, Clint (31 de ene. de 2013). «Why Russ is wrong». url: http://xana.scru.org/xana2/bamamba/consensed/.

Allbery, Russ (30 de ene. de 2013). «Consensus failure». url: http://www.eyrie.org/~eagle/journal/2013-01/026.html.

Armstrog, Don (14 de nov. de 2014). «Adding a newcomer tag to the BTS». url: http://www.donarmstrong.com/posts/newcomer_bts_tag/.

Berg, Cristoph (7 de jun. de 2006). «Post-DebConf Woes». url: http://www.df7cb.de/blog/2006/Post-Debconf.html.

Brockmeier, Joe (2 de abr. de 2010). «The role of the Debian ftpmasters». *LWN.net*. url: http://lwn.net/Articles/381667/.

Bromberger, James E. (13 de feb. de 2012). «Debian Wheezy: US 19 Billion. Your price... FREE!» url: https://blog-origin.james.rcpt.to/2012/02/13/debian-wheezy-us19-billion-your-price-free/.

Byfield, Bruce (7 de ene. de 2005). «Hot Babe and Debian: A test case for community standards in free software». *Linux.com*. url: http://www.linux.com/news/hot-babe-and-debian-test-case-community-standards-free-software.

Byfield, Bruce (21 de sep. de 2006). «Proposal to fund Debian reveals debate about developers' motivations». *Linux.com*. url: https://www.linux.com/news/proposal-fund-debian-reveals-debate-about-developers-motivations.

— (3 de ene. de 2007). «Dunc-Tank continues to make splash». *Linux.com*. url: http://www.linux.com/news/dunc-tank-continues-make-splash.

Coleman, Gabriella (s.f.). «Debian History Roundtable Discussion. History Roundtable Discussion @ Debconf4, june 2004, Porto Alegre, Brazil» (). url: http://gabriellacoleman.org/debian-history-roundtable-discussion/.

Corbet, Jonathan (2005). «The Kernel and binary firmware». *LWN.net*. url: http://lwn.net/Articles/130696/.

— (5 de nov. de 2013). «Which init system for Debian?» *LWN.net*. url: http://lwn.net/Articles/572805/.

— (14 de sep. de 2014). «How to enforce Debian's code of conduct». *LWN.net*. url: http://lwn.net/Articles/611317/.

Ferrari, Martin (8 de ago. de 2012). «Debian whois: a proposal to increase visibility and recognition of contributors». url: http://blog.tincho.org/posts/whois/.

Goerzen, John (30 de mayo de 2009). «Free Software enforcing DRM?!» url: http://changelog.complete.org/archives/1042-free-software-enforcing-drm.

Hertzog, Raphaël (27 de oct. de 2008). «Debian membership reform». url: http://raphaelhertzog.com/2008/10/27/debian-membership-reform/.

— (24 de ene. de 2011a). «Debian is eating its own dog food more than ever». url: http://raphaelhertzog.com/2011/01/24/debian-is-eating-its-own-dog-food-more-than-ever/.

— (30 de jun. de 2011b). «How to start contributing to Debian?» url: http://raphaelhertzog.com/2011/06/30/how-to-start-contributing-to-debian/.

— (8 de ago. de 2012a). «How to use quilt to manage patches in Debian packages». url: http://raphaelhertzog.com/2012/08/08/how-to-use-quilt-to-manage-patches-in-debian-packages/.

— (6 de abr. de 2012b). «People Behind Debian: Francesca Ciceri, Member of the Debian Press & Publicity Teams». url: http://raphaelhertzog.com/2012/04/06/people-behind-debian-francesca-ciceri-member-of-debian-press-publicity-teams/.

— (23 de mar. de 2012c). «People behind Debian: Jörg Jaspert, FTPmaster, Debian Account Manager, and more». url: http://raphaelhertzog.com/2012/03/23/people-behind-debian-joerg-jaspert/.

— (s.f.). «Freexian's reports about Debian Long Term Support» (). url: http://raphaelhertzog.com/tag/Freexian+LTS/.

Hess, Joey (25 de mar. de 2006). «Ending the tyranny of unix permissions». url: http://joeyh.name/blog/entry/ending_the_tyranny_of_unix_permissions/.

— (23 de oct. de 2008). «anatomy of an atrocious announcement». url: http://joeyh.name/blog/entry/anatomy_of_an_atrocious_announcement/.

Hill, Benjamin Mako (7 de ago. de 2005). «To Fork or Not To Fork: Lessons From Ubuntu and Debian». url: https://mako.cc/writing/to_fork_or_not_to_fork.html.

— (14 de jul. de 2009). «Taking a Principled Position on Software Freedom». url: http://mako.cc/copyrighteous/taking-a-principled-position-on-software-freedom.

— (4 de jun. de 2010). «Free Software Needs Free Tools». url: http://mako.cc/writing/hill-free_tools.html.

— (2012). «Problems and Strategies in Financing Voluntary Free Software Projects (revision 0.2.1)». url: http://mako.cc/writing/funding_volunteers/funding_volunteers.html.

Jaspert, Joerg (28 de jul. de 2007a). «The DM GR». url: http://blog.ganneff.de/blog/2007/07/27/the-dm-gr.html.

— (14 de jul. de 2007b). «WaT - Where Are They?» url: http://blog.ganneff.de/2007/07/wat-where-are-they.html.

— (24 de oct. de 2008). «Developer status». url: http://blog.ganneff.de/2008/10/developer-status.html.

Kahn Gillmor, Daniel (8 de mayo de 2015). «Cheers to audacity!» url: http://debian-administration.org/users/dkg/weblog/114.

Kerrisk, Michael (10 de abr. de 2013). «Legal issues from a radical community angle». *LWN.net*. url: http://lwn.net/Articles/546411/.

Krafft, Martin F. (24 de mayo de 2006a). «A non-official ID at the keysigning». url: http://madduck.net/blog/2006.05.24:tr-id-at-keysigning/.

— (27 de mayo de 2006b). «Aftereffects of the keysigning experiment». url: http://madduck.net/blog/2006.05.27:keysigning-again/.

Kuhn, Bradley M. (14 de ene. de 2010). «Back Home, with Debian!» url: http://ebb.org/bkuhn/blog/2010/01/14/ubuntu-debian.html.

Ledkov, Dimitri John (27 de ene. de 2013). «Thoughts on Debian package policies». url: http://blog.surgut.co.uk/2013/01/thoughts-on-debian-package-policies.html.

Mallach, Jordi (24 de nov. de 2009). «Ten years as a Debian Maintainer». url: http://oskuro.net/blog/freesoftware/ten-years-debian-maintainer-2009-11-24-23-47.

— (7 de ago. de 2014). «A pile of reasons why GNOME should be Debian jessie's default desktop environment». url: http://oskuro.net/blog/freesoftware/gnome-as-default-jessie-desktop-2014-08-07-23-58.

Matzan, Jem (5 de jun. de 2004). «The gift economy and free software». *Linux.com*. url: http://www.linux.com/news/gift-economy-and-free-software.

Mones, Ricardo (5 de feb. de 2014). «Fixing partridge eggs with industrial duct tape». url: http://mones.livejournal.com/109209.html.

Nussbaum, Lucas (12 de ago. de 2008a). «Debian's Freeze». url: http://www.lucas-nussbaum.net/blog/?p=305.

— (24 de oct. de 2008b). «Developer status: problems and solutions». url: http://www.lucas-nussbaum.net/blog/?p=317.

— (24 de jun. de 2009). «On the New Maintainer Process». url: http://www.lucas-nussbaum.net/blog/?p=349.

— (29 de sep. de 2010a). «How to demotivate people (RE: Making Debian Responsible ofr its actions)». url: http://www.lucas-nussbaum.net/blog/?p=582.

— (12 de sep. de 2010b). «Ruby Packaging in Debian and Ubuntu: Mythbusting and FAQ». url: http://www.lucas-nussbaum.net/blog/?p=566.

— (2 de ene. de 2011). «Giving up on Ruby packaging». url: http://www.lucas-nussbaum.net/blog/?p=617.

Parrish, Allison (10 de jul. de 2016). «Programming is Forgetting: Toward a New Hacker Ethic». url: http://opentranscripts.org/transcript/programming-forgetting-new-hacker-ethic/.

Pearlmutter, Barak A. (s.f.). «DFSG and Software License FAQ (Draft)» (). url: http://people.debian.org/~bap/dfsg-faq.

Perens, Bruce (15 de jun. de 1999). «About the"Open Source" Trademark and the Open Source Definition». *Linuxtoday.com*. url: http://www.linuxtoday.com/developer/1999061503710NWSM.

ris (27 de sep. de 2006). «Debian + Dunc-Tank.org = dissent». *LWN.net*. url: http://lwn.net/Articles/201488/.

Schulze, Joey (7 de feb. de 2007). «Does Debian need a Social Committee?» *LWN.net*. url: http://lwn.net/Articles/221077/.

Sobol, Rebecca (29 de jul. de 2009). «A tale of two shells: bash or dash». *LWN.net*. url: http://lwn.net/Articles/343924/ (visitado 06-08-2013).

Stallman, Richard (2008). «Thee Free Software Community After 20 Years: With great but incomplete success, what now?» url: http://www.gnu.org/philosophy/use-free-software.html.

— (2015). «Free Software Is Even More Important Now». url: http://www.gnu.org/philosophy/free-software-even-more-important.html.

Tay, Liz (24 de ene. de 2007). «Dunc-Tank: Success or failure?» *Computerworld.com*. url: http://www.computerworld.com.au/article/172729/dunc-tank_success_failure.

Towns, Anthony (12 de abr. de 2006). «Developers and Maintainers». url: http://www.erisian.com.au/wordpress/2006/04/12/developers-and-maintainers.

Varghese, Sam (31 de mar. de 2015). «Surviving systemd: Lucas Nussbaum satisfied with init system outcome». *iTWire*. url: http://www.itwire.com/business-it-news/open-source/67512-surviving-systemd-lucas-nussbaum-satisfied-with-init-system-outcome.

Verhelst, Wouter (23 de abr. de 2008). «Planet installations». url: http://grep.be/blog/en/life/debian/planet.

Williams, Neil (dic. de 2009). «When firmware is not software». url: http://www.linux.codehelp.co.uk/serendipity/index.php?/archives/148-When-firmware-is-not-software.html.

Willis, Nathan (26 de ago. de 2015). «Copyright assignment and license enforcement for Debian». *LWN.net*. url: http://lwn.net/Articles/655009/.

Wirzenius, Lars (8 de abr. de 2012). «What is a Linux distribution?» url: http://blog.liw.fi/posts/what-is-a-distro/.

— (17 de feb. de 2018). «What is Debian all about, really? Or: friction, packaging complex applications». url: http://blog.liw.fi/posts/2018/02/17/what_is_debian_all_about_really_or_friction_packaging_complex_applications/.

Wolf, Gunnar (30 de jul. de 2007). «On the Debian Maintainers GR». url: http://gwolf.org/blog/show/On-the-Debian-Maintainers-GR.

— (29 de sep. de 2010). «Ruby dissonance with Debian, again». url: http://gwolf.org/blog/ruby-dissonance-debian-again.

— (6 de ene. de 2011a). «Back with Ruby on Debian polemics». url: http://gwolf.org/blog/back-ruby-debian-polemics.

— (18 de ago. de 2013). «After Debian's 20th birthday, everybody's life gets back to normality...» url: http://gwolf.org/blog/after-debians-20th-birthday-everybodys-life-gets-back-normality.

Wolf, Gunnar (5 de jun. de 2014). «What defines an identity?» url: http://gwolf.org/node/3928.

Wülfing, Britta (9 de mayo de 2008). «Anthony Towns Withdraws from Debian Projects». *Linux Magazine*. url: http://www.linux-magazine.com/Online/News/Anthony-Towns-Withdraws-from-Debian-Projects.

Zacchiroli, Stefano (13 de mayo de 2010a). «Collaboration with Ubuntu from the Debian point of view». url: http://upsilon.cc/~zack/talks/2010/201005-uds.pdf.

— (jul. de 2010b). «DebConf BoF HOWTO». url: http://upsilon.cc/~zack/blog/posts/2010/07/DebConf_BoF_HOWTO/.

— (dic. de 2010c). «Debian 6.0 Squeeze to be released with completely free Kernel(s)». url: http://upsilon.cc/~zack/blog/posts/2010/12/squeeze_your_non-free_firmware_away/.

— (mayo de 2010d). «document your team - redux». url: http://upsilon.cc/~zack/blog/posts/2010/05/document_your_team_-_redux/.

— (mayo de 2010e). «Eclectic paper on the Ultimate Debian Database». url: http://upsilon.cc/~zack/blog/posts/2010/05/UDD_-_consolidating_bazaar_metadata_for_QA_and_data_mining/.

— (jun. de 2010f). «How to have a (Debian) summit without turning into a secret cabal». url: http://upsilon.cc/~zack/blog/posts/2010/06/debian_meeting_guidelines_RFC/.

— (ene. de 2010g). «Kuhn on Debian, Ubuntu, and the culture of freedom». url: http://upsilon.cc/~zack/blog/posts/2010/01/kuhn_on_debian_ubuntu_and_the_culture_of_freedom/.

— (ene. de 2010h). «RCBW: let's fix one Debian RC bug per-day». url: http://upsilon.cc/~zack/hacking/debian/rcbw/.

— (jul. de 2011a). «16 months of Debian Sprints». url: http://upsilon.cc/~zack/blog/posts/2011/07/16_months_of_debian_sprints/.

— (26 de ago. de 2011b). «Debian in Context. Distributions, Upstreams, and Downstreams». url: http://upsilon.cc/~zack/talks/2011/20110826-ghm.pdf.

— (mar. de 2012a). «debian contributions to the linux kernel». url: http://upsilon.cc/~zack/blog/posts/2012/03/debian_contributions_to_the_linux_kernel/.

— (21 de jun. de 2012b). «Grassroots Free Software In-Depth Case Study: Debian». url: http://upsilon.cc/~zack/talks/2012/20120621-iiosss.pdf.

— (4 de abr. de 2013). «Legal issues from a radical community angle». url: http://upsilon.cc/~zack/talks/2013/20130404-fsfe-legal.pdf.
Zini, Enrico (30 de mar. de 2006). «Debian Community Guidelines». url: http://people.debian.org/~enrico/dcg/.
— (3 de sep. de 2013a). «A moving commit». url: http://www.enricozini.org/blog/2013/debian/a-moving-commit/.
— (19 de ago. de 2013b). «Debian Contributors right after DebConf». url: http://www.enricozini.org/blog/2013/debian/debian-contributors-right-after-debconf/.
Zini, Enrico y Francesca Ciceri (12 de jul. de 2012). «More diversity in Debian skills». url: http://www.enricozini.org/blog/2012/debian/more-diversity-in-skills/.
Zobel-Helas, Martin (22 de jun. de 2011). «How YOU can help Debian!» url: http://blog.zobel.ftbfs.de/2011/06/22/how_you_can_help_debian_1.html.

Páginas *web* y listas de correo

Alioth-lists.debian.net (mar. de 2016). «[Pkg-owncloud-maintainers] About no more ownCloud in Debian». url: http://alioth-lists.debian.net/pipermail/pkg-owncloud-maintainers/2016-March/002899.html.
Bits.debian.org (2 de feb. de 2018). «Debian welcomes its Outreachy interns». url: http://bits.debian.org/2018/02/welcome-outreachy-interns-2017-2018.html.
Boardgamegeek.com (s.f.). «Mao». url: http://boardgamegeek.com/boardgame/4213/mao.
Bugs.debian.org (s.f.[a]). «Debian bug tracking system – Página principal». url: http://bugs.debian.org.
— (s.f.[b]). «tech-ctte: Decide which init system to default to in Debian». url: http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=727708.
— (s.f.[c]). «Debian Archived Bug report logs: Bugs in package tech-ctte». url: http://bugs.debian.org/cgi-bin/pkgreport.cgi?archive=both;package=tech-ctte.
— (s.f.[d]). «Please decide on Python interpreter packages maintainership». url: http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=573745.

Bugs.debian.org (s.f.[e]). «Re: Bug 573745: ping». url: http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=573745#330.

— (s.f.[f]). «Please decide on Python interpreter packages maintainership (msg 92)». url: http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=573745;msg=92.

— (s.f.[g]). «Please decide on the ”ownership” of the developers reference». url: http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=436093.

— (s.f.[h]). «ftp.debian.org: Is AGPLv3 DFSG-free?» url: http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=495721.

— (s.f.[i]). «ITP: nyancat – Terminal-based Pop Tart Cat animation». url: http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=661565.

— (s.f.[j]). «ITP: hot-babe – erotic graphical system activity monitor». url: http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=283578.

Burningman.org (s.f.). «Burning Man». url: http://burningman.org/.

Communitywiki.org (s.f.). «Doocracy». url: http://communitywiki.org/wiki/DoOcracy.

Contributors.debian.org (s.f.[a]). «Debian Contributors». url: http://contributors.debian.org/.

— (s.f.[b]). «Debian Contributors data sources». url: http://contributors.debian.org/sources/.

Cpan.org (s.f.). «Comprehensive Perl Archive Network». url: http://www.cpan.org/.

Cran.r-project.org (s.f.). «The Comprehensive R Archive Network». url: http://cran.r-project.org/.

Creativecommons.org (s.f.). «Creative Commons – Página principal». url: http://creativecommons.org/.

Ctan.org (s.f.). «Comprehensive TEX Archive Network». url: http://ctan.org/.

Db.debian.org (s.f.[a]). «Developer Machines». url: http://db.debian.org/machines.cgi.

— (s.f.[b]). «Db Debian». url: http://db.debian.org/.

Debconf-discuss (s.f.). «Debconf discuss list – Página principal». url: http://lists.debian.org/debconf-discuss/.

— (mayo de 2006). «[Debconf-discuss] Please revoke your signatures from Martin Kraff's keys». url: http://lists.debian.org/debconf-discuss/2006/05/msg00172.html.

— (jun. de 2009). «[Debconf-discuss] GPG keysigning?» url: http://lists.debian.org/debconf-discuss/2009/06/msg00058.html.
Debconf.org (s.f.[a]). «DebConf – The Debian Developer Conference». url: http://www.debconf.org/.
— (s.f.[b]). «DebConf resources». url: http://www.debconf.org/resources.shtml.
— (s.f.[c]). «DebConf9. 11 Lightning Talks». url: http://penta.debconf.org/dc9_schedule/events/461.en.html.
Debconf16.debconf.org (s.f.[a]). «2 Years of Work of Paid Contributors in the Debian LTS Project». url: http://debconf16.debconf.org/talks/40/.
— (s.f.[b]). «Using Debian Money to Fund Debian Projects». url: http://debconf16.debconf.org/talks/41/.
— (s.f.[c]). «The Supreme Court of DFSG-Free?» url: http://debconf16.debconf.org/talks/38/.
Debian-announce (s.f.). «Debian announce list – Página principal». url: http://lists.debian.org/debian-announce/.
— (2009). «Debian decides to adopt time-based release freezes». url: http://lists.debian.org/debian-announce/2009/msg00009.html.
Debian-ctte (s.f.). «Debian ctte list – Página principal». url: http://lists.debian.org/debian-ctte/.
— (mar. de 2011a). «Bug 573745: ping». url: http://lists.debian.org/debian-ctte/2011/03/msg00002.html.
— (jul. de 2012a). «debian-ctte git repository». url: http://lists.debian.org/debian-ctte/2012/07/msg00275.html.
— (jul. de 2012b). «Draft GR for permitting private discussion». url: http://lists.debian.org/debian-ctte/2012/07/msg00011.html.
— (jul. de 2012c). «Re: Draft GR for permitting private discussion». url: http://lists.debian.org/debian-ctte/2012/07/msg00121.html.
— (jul. de 2012d). «Re: Draft GR for permitting private discussion». url: http://lists.debian.org/debian-ctte/2012/07/msg00099.html.
— (mar. de 2010a). «Bug 573745: Please decide on Python interpreter packages maintainership». url: https://lists.debian.org/debian-ctte/2010/03/msg00012.html.
— (mayo de 2010b). «Re: Bug 573745: python maintainance: next steps». url: http://lists.debian.org/debian-ctte/2010/05/msg00013.html.
— (mar. de 2011b). «Bug 573745: ping». url: http://lists.debian.org/debian-ctte/2011/03/msg00010.html.

Debian-devel (s.f.). «Debian devel list – Página principal». url: http://lists.debian.org/debian-devel/.
— (jul. de 2012a). «“Hijacking packages for fun and profit” BoF at DebConf». url: http://lists.debian.org/debian-devel/2012/07/msg00540.html.
— (mayo de 2010). «Too much disruptive NMUs». url: http://lists.debian.org/debian-devel/2010/05/msg00760.html.
— (oct. de 2012b). «[SUMMARY/PROPOSAL] Orphaning another maintainer’s packages». url: http://lists.debian.org/debian-devel/2012/10/msg00469.html.
— (oct. de 2012c). «[PROPOSAL v2] Orphaning another maintainer’s packages». url: http://lists.debian.org/debian-devel/2012/10/msg00618.html.
— (abr. de 2016a). «Re: Overall bitrot, package reviews and fast(er) unmaintained package removals». url: http://lists.debian.org/debian-devel/2016/04/msg00105.html.
— (abr. de 2018). «salvage != hijacking». url: http://lists.debian.org/debian-devel/2018/04/msg00311.html.
— (ago. de 2014a). «Reverting to GNOME for jessie’s default desktop». url: http://lists.debian.org/debian-devel/2014/08/msg00129.html.
— (ago. de 2014b). «Re: Reverting to GNOME for jessie’s default desktop». url: http://lists.debian.org/debian-devel/2014/08/msg00140.html.
— (ago. de 2014c). «Re: Reverting to GNOME for jessie’s default desktop». url: http://lists.debian.org/debian-devel/2014/08/msg00365.html.
— (sep. de 2016b). «Re: Debian does not have customers, but users». url: http://lists.debian.org/debian-devel/2016/09/msg00418.html.
— (mayo de 2009). «Bug 531221: okular: Arbitrarily enforces DRM». url: http://lists.debian.org/debian-devel/2009/05/msg00879.html.
— (ago. de 2014d). «Standardizing the layout of git packaging repositories». url: http://lists.debian.org/debian-devel/2014/08/msg00499.html.
— (sep. de 2016c). «Re: Debian does not have customers». url: http://lists.debian.org/debian-devel/2016/09/msg00456.html.
— (sep. de 2016d). «Re: Debian does not have customers». url: http://lists.debian.org/debian-devel/2016/09/msg00463.html.
— (feb. de 2012d). «Bug 661565: ITP: nyancat – Terminal-based Pop Tart Cat animation». url: http://lists.debian.org/debian-devel/2012/02/msg01184.html.

— (jun. de 2012e). «Maintainers, teams, Uploaders, DMs, DDs, etc.» url: http://lists.debian.org/debian-devel/2012/06/msg00479.html.
— (oct. de 2014e). «Re: Determining, ad hoc, whether someone is a DD». url: http://lists.debian.org/debian-devel/2014/10/msg00354.html.
— (oct. de 2014f). «Determining, ad hoc, whether someone is a DD». url: http://lists.debian.org/debian-devel/2014/10/msg00349.html.
— (jun. de 2016e). «Keysigning via Video Conferencing». url: http://lists.debian.org/debian-devel/2016/06/msg00297.html.
— (jun. de 2016f). «Re: Keysigning via Video Conferencing». url: http://lists.debian.org/debian-devel/2016/06/msg00386.html.
Debian-devel-announce (s.f.). «Debian Devel Announce list – Página principal». url: http://lists.debian.org/debian-devel-announce/.
— (oct. de 2012a). «[CTTE 573745] Maintainership of python packages in Debian». url: http://lists.debian.org/debian-devel-announce/2012/10/msg00000.html.
— (mar. de 2010a). «Bits from an FTP Master: New team member, need more volunteers, (kind of) todo list». url: http://lists.debian.org/debian-devel-announce/2010/03/msg00003.html.
— (mar. de 2003). «Introducing Alioth: SourceForge for Debian». url: http://lists.debian.org/debian-devel-announce/2003/03/msg00024.html.
— (oct. de 2010b). «Debian sprint program». url: http://lists.debian.org/debian-devel-announce/2010/10/msg00014.html.
— (oct. de 2008). «Developer Status». url: http://lists.debian.org/debian-devel-announce/2008/10/msg00005.html.
— (oct. de 1999). «New maintainer situation». url: http://lists.debian.org/debian-devel-announce/1999/10/msg00003.html.
— (oct. de 2011). «Bits from the New Maintainer^WMember process». url: http://lists.debian.org/debian-devel-announce/2011/10/msg00004.html.
— (feb. de 2005). «Bits from the DAMs». url: http://lists.debian.org/debian-devel-announce/2005/02/msg00003.html.
— (dic. de 2007). «Bits from the MIA team». url: http://lists.debian.org/debian-devel-announce/2007/12/msg00005.html.
— (nov. de 2009). «Bits from the NM people». url: http://lists.debian.org/debian-devel-announce/2009/11/msg00005.html.
— (mayo de 2010c). «Bits from the NM process: advocacy messages». url: http://lists.debian.org/debian-devel-announce/2010/05/msg00003.html.

Debian-devel-announce (nov. de 2010d). «Guidelines for applying as non-uploading DD». url: http://lists.debian.org/debian-devel-announce/2010/11/msg00000.html.
— (mayo de 2012b). «bits from the NM process: advocacy, no more AM reports, AMs needed». url: http://lists.debian.org/debian-devel-announce/2012/05/msg00007.html.
— (jun. de 2016a). «Changes in the New Member process». url: http://lists.debian.org/debian-devel-announce/2016/06/msg00003.html.
— (ago. de 2016b). «Bits from Debian New Members». url: http://lists.debian.org/debian-devel-announce/2016/08/msg00007.html.
— (ago. de 2017). «Bits from the New Member Process». url: http://lists.debian.org/debian-devel-announce/2017/08/msg00009.html.
— (dic. de 2013). «Debian Contributors». url: http://lists.debian.org/debian-devel-announce/2013/12/msg00009.html.
— (mar. de 2014). «Debian Contributors want YOUr data source!» url: http://lists.debian.org/debian-devel-announce/2014/03/msg00005.html.
Debian-devel-spanish (s.f.). «Debian devel spanish list – Página principal». url: http://lists.debian.org/debian-devel-spanish/.
— (abr. de 2008). «Re: Nuevo». url: http://lists.debian.org/debian-devel-spanish/2008/04/msg00026.html.
Debian-l10n-spanish (s.f.). «Spanish Localization list – Página principal». url: http://lists.debian.org/debian-l10n-spanish/.
— (sep. de 2010a). «Re: Presentación». url: http://lists.debian.org/debian-l10n-spanish/2010/09/msg00239.html.
— (oct. de 2010b). «Presentación Pablo». url: http://lists.debian.org/debian-l10n-spanish/2010/10/msg00376.html.
Debian-legal (s.f.). «Debian legal list – Página principal». url: http://lists.debian.org/debian-legal/.
— (nov. de 2008). «AGPL and Debian». url: http://lists.debian.org/debian-legal/2008/11/msg00097.html.
Debian-newmaint (s.f.). «Debian newmaint list – Página principal». url: http://lists.debian.org/debian-newmaint/.
— (sep. de 2011a). «on the “M” of “NM”». url: http://lists.debian.org/debian-newmaint/2011/09/msg00071.html.
— (mayo de 2006). «Proposal: The future of the Debian NM process». url: http://lists.debian.org/debian-newmaint/2006/05/msg00015.html.

— (jul. de 2009). «Notes from the NM AM FD DAM BOF». url: http://lists.debian.org/debian-newmaint/2009/07/msg00034.html.
— (jul. de 2012). «one of the many reasons DebConfs are great (Re: DM application of Ian Campbell)». url: http://lists.debian.org/debian-newmaint/2012/07/msg00032.html.
— (mayo de 2011b). «AM report for Francesca Ciceri <madamezou@yahoo.it>». url: http://lists.debian.org/debian-newmaint/2011/05/msg00022.html.
Debian-project (s.f.). «Debian Project list – Página principal». url: http://lists.debian.org/debian-project/.
— (feb. de 2007a). «Re: Bits from the DPL: DSA and buildds and DAM, oh my!» url: http://lists.debian.org/debian-project/2007/02/msg00214.html.
— (sep. de 2009a). «Re: Distributing software written by hostile upstream developers». url: http://lists.debian.org/debian-project/2009/09/msg00072.html.
— (ene. de 2007b). «Social Committee proposal». url: http://lists.debian.org/debian-project/2007/01/msg00063.html.
— (mayo de 2010a). «debian-private declassification team (looking for one)». url: http://lists.debian.org/debian-project/2010/05/msg00105.html.
— (oct. de 2010b). «Re: QTS (was Re: user support - Shapado instance for Debian)». url: http://lists.debian.org/debian-project/2010/10/msg00049.html.
— (jul. de 2009b). «Re: Debian decides to adopt time-based release freezes». url: http://lists.debian.org/debian-project/2009/07/msg00148.html.
— (dic. de 2016a). «Replace the TC power to depose maintainers». url: http://lists.debian.org/debian-project/2016/12/msg00001.html.
— (dic. de 2016b). «Re: Replace the TC power to depose maintainers». url: http://lists.debian.org/debian-project/2016/12/msg00009.html.
— (dic. de 2016c). «Re: Replace the TC power to depose maintainers». url: http://lists.debian.org/debian-project/2016/12/msg00013.html.
— (dic. de 2016d). «Maintainerless Hive-Mind? (was Re: Replace the TC power to depose maintainers)». url: http://lists.debian.org/debian-project/2016/12/msg00086.html.
— (ago. de 2013). «Survey of new contributors – results». url: http://lists.debian.org/debian-project/2013/08/msg00011.html.
— (sep. de 2011a). «report from GNU hackers meeting 2011». url: http://lists.debian.org/debian-project/2011/09/msg00004.html.

Debian-project (mayo de 2010c). «Debian-Ubuntu relationship: bits from UDS». url: http://lists.debian.org/debian-project/2010/05/msg00084.html.
— (mayo de 2010d). «Debian-Ubuntu relationship: poll summary». url: http://lists.debian.org/debian-project/2010/05/msg00083.html.
— (oct. de 2011b). «relationship with Ubuntu - call for feedback». url: http://lists.debian.org/debian-project/2011/10/msg00057.html.
— (mar. de 2014). «clarify FTP master delegation?» url: http://lists.debian.org/debian-project/2014/03/msg00134.html.
— (nov. de 2010e). «No general political content on Planet». url: http://lists.debian.org/debian-project/2010/11/msg00010.html.
— (nov. de 2010f). «Please draft a policy for planet.debian.org». url: http://lists.debian.org/debian-project/2010/11/msg00084.html.
— (nov. de 2010g). «Re: No general political content on Planet». url: http://lists.debian.org/debian-project/2010/11/msg00108.html.
— (nov. de 2010h). «Re: Please draft a policy for planet.debian.org». url: http://lists.debian.org/debian-project/2010/11/msg00114.html.
— (ago. de 2009c). «Summary of the debian-devel BoF at Debconf9». url: http://lists.debian.org/debian-project/2009/08/msg00278.html.
— (oct. de 1999). «New maintainer proposal». url: http://lists.debian.org/debian-project/1999/10/msg00003.html.
— (jun. de 2009d). «debian-project Jun 2009 by thread». url: http://lists.debian.org/debian-project/2009/06/threads.html.
— (jul. de 2009e). «Re-thinking Debian membership - take #1: inactivity». url: http://lists.debian.org/debian-project/2009/07/msg00067.html.
— (ago. de 2009f). «Re-thinking Debian membership - take #1: inactivity - status update». url: http://lists.debian.org/debian-project/2009/08/msg00018.html.
— (dic. de 2009g). «Re-thinking Debian membership - take #1: inactivity - getting implemented». url: http://lists.debian.org/debian-project/2009/12/msg00020.html.
— (oct. de 2008a). «Re: Developer Status». url: http://lists.debian.org/debian-project/2008/10/msg00046.html.
— (oct. de 2008b). «Re-thinking Debian membership». url: http://lists.debian.org/debian-project/2008/10/msg00145.html.
— (oct. de 2008c). «Re: Developer Status». url: http://lists.debian.org/debian-project/2008/10/msg00049.html.

— (jun. de 2009h). «Re: DAM queues processing». url: http://lists.debian.org/debian-project/2009/06/msg00053.html.
— (mar. de 2009i). «Debian Membership». url: http://lists.debian.org/debian-project/2009/03/msg00053.html.
— (abr. de 2012a). «Re: xth wrap-up about statement on diversity, statement may be issued without general resolution». url: http://lists.debian.org/debian-project/2012/04/msg00088.html.
— (mar. de 2012b). «Diversity statement for the Debian Project». url: http://lists.debian.org/debian-project/2012/03/msg00048.html.
— (jun. de 2009j). «DAM and NEW queues processing». url: http://lists.debian.org/debian-project/2009/06/msg00024.html.
Debian-vote (s.f.). «Debian vote list – Página principal». url: http://lists.debian.org/debian-vote/.
— (abr. de 2017). «having public irc logs?» url: http://lists.debian.org/debian-vote/2017/04/msg00022.html.
— (mar. de 2013). «Re: Your opinion on Debian Maintainer status». url: http://lists.debian.org/debian-vote/2013/03/msg00200.html.
— (mar. de 2010a). «Re: Question about membership». url: http://lists.debian.org/debian-vote/2010/03/msg00294.html.
— (mar. de 2010b). «Re: Question about membership.» url: http://lists.debian.org/debian-vote/2010/03/msg00309.html.
— (oct. de 2008a). «Call for seconds: Suspension of the changes of the Project's membership procedures». url: http://lists.debian.org/debian-vote/2008/10/msg00103.html.
— (oct. de 2008b). «DAM has no competency to make changes to membership structure (was: Call for seconds: Suspension of the changes of the Project's membership procedures.)» url: http://lists.debian.org/debian-vote/2008/10/msg00183.html.
— (dic. de 2008c). «Re: Results for Project membership procedures». url: http://lists.debian.org/debian-vote/2008/12/msg00166.html.
— (mar. de 2009). «All candidates: Membership procedures». url: http://lists.debian.org/debian-vote/2009/03/msg00088.html.
— (sep. de 2010c). «GR: welcome non-packaging contributors as Debian project members». url: http://lists.debian.org/debian-vote/2010/09/msg00004.html.

Debian-vote (sep. de 2010d). «Re: GR: welcome non-packaging contributors as Debian project members». url: http://lists.debian.org/debian-vote/2010/09/msg00052.html.
— (sep. de 2010e). «Re: GR: welcome non-packaging contributors as Debian project members». url: http://lists.debian.org/debian-vote/2010/09/msg00017.html.
— (sep. de 2010f). «Re: GR: welcome non-packaging contributors as Debian project members». url: http://lists.debian.org/debian-vote/2010/09/msg00028.html.
Debian.org (s.f.[a]). «Página *web* principal de Debian». url: http://debian.org.
— (s.f.[b]). «Debian bug tracking system». url: http://www.debian.org/Bugs.
— (s.f.[c]). «About Debian». url: http://www.debian.org/intro/about.
— (s.f.[d]). «¿Cómo puede ayudar a Debian?» url: http://www.debian.org/intro/help.
— (s.f.[e]). «Cómo puede ayudar». url: http://www.debian.org/devel/join/.
— (s.f.[f]). «How You Can Join». url: http://www.debian.org/devel/join/index.en.html.
— (s.f.[g]). «Platform for Stefano Zacchiroli». url: http://www.debian.org/vote/2010/platforms/zack.
— (s.f.[h]). «Mailing Lists». url: http://www.debian.org/MailingLists/.
— (s.f.[i]). «Debian Constitution». url: http://www.debian.org/devel/constitution.
— (s.f.[j]). «Debian Technical Committee». url: http://www.debian.org/devel/tech-ctte.
— (s.f.[k]). «Debian Code of Conduct». url: http://www.debian.org/code_of_conduct.
— (s.f.[l]). «Ports». url: http://www.debian.org/ports/.
— (s.f.[m]). «Debian's Organizational Structure». url: http://www.debian.org/intro/organization.
— (s.f.[n]). «Debian Project Leader». url: http://www.debian.org/devel/leader.
— (s.f.[o]). «The Debian Project Secretary». url: http://www.debian.org/devel/secretary.
— (s.f.[p]). «Debian Machine Usage Policies». url: http://www.debian.org/devel/dmup.
— (s.f.[q]). «Packages». url: http://www.debian.org/distrib/packages.
— (s.f.[r]). «Debian Trademarks». url: http://www.debian.org/trademark.

— (s.f.[s]). «Consultants». url: http://www.debian.org/consultants/.
— (s.f.[t]). «Information for Debian consultants». url: http://www.debian.org/consultants/info.
— (s.f.[u]). «What Does Free Mean? or What do you mean by Free Software?» url: http://www.debian.org/intro/free.
— (s.f.[v]). «License information». url: http://www.debian.org/legal/licenses/.
— (s.f.[w]). «Debian Position on Software Patents». url: http://www.debian.org/legal/patent.
— (s.f.[x]). «Community Distribution Patent Policy FAQ». url: http://www.debian.org/reports/patent-faq.
— (s.f.[y]). «Debian New Members Corner». url: http://www.debian.org/devel/join/newmaint.
— (s.f.[z]). «Step 3: Philosophy and Procedures». url: http://www.debian.org/devel/join/nm-step3.
— (s.f.[aa]). «Advocating a prospective member». url: http://www.debian.org/devel/join/nm-advocate.
— (s.f.[ab]). «Diversity Statement». url: http://www.debian.org/intro/diversity.
— (s.f.[ac]). «The Debian Women Project». url: http://www.debian.org/women/.
— (s.f.[ad]). «About Debian Women». url: http://www.debian.org/women/about.
— (s.f.[ae]). «Debian Social Contract». url: http://www.debian.org/social_contract.
— (s.f.[af]). «Debian Free Software Guidelines». url: http://www.debian.org/social_contract#guidelines.

Debian.org/doc (s.f.[a]). «Documentation». url: https://www.debian.org/doc/.

— (s.f.[b]). «The Debian GNU/Linux FAQ . Chapter 7 - Basics of the Debian package management system». url: http://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics.
— (s.f.[c]). «A Brief History of Debian». url: http://www.debian.org/doc/manuals/project-history/.
— (s.f.[d]). «Debian Policy Manual». url: http://www.debian.org/doc/debian-policy/.
— (s.f.[e]). «Debian Developer's Reference». url: https://www.debian.org/doc/manuals/developers-reference/.
— (s.f.[f]). «Debian New Maintainers' Guide». url: http://www.debian.org/doc/manuals/maint-guide/.

Debian.org/doc (s.f.[g]). «The Debian GNU/Linux FAQ. Chapter 8 - The Debian package management tools». url: http://www.debian.org/doc/manuals/debian-faq/ch-pkgtools.
— (s.f.[h]). «Machine-readable debian/copyright file». url: http://www.debian.org/doc/packaging-manuals/copyright-format/1.0/.
— (s.f.[i]). «The Debian Manifesto». url: http://www.debian.org/doc/manuals/project-history/ap-manifesto.en.html.
Debian.org/News (s.f.). «News – Página principal». url: http://www.debian.org/News/index.en.html.
— (15 de dic. de 2010). «Debian 6.0 Squeeze to be released with completely free Linux Kernel». url: http://www.debian.org/News/2010/20101215.
— (1 de mar. de 2013). «Discover Debian's hassle-free trademarks, use them to promote Debian». url: http://www.debian.org/News/2013/20130301.
— (17 de ago. de 2015). «Debian and Software Freedom Conservancy announce Copyright Aggregation Project». url: http://www.debian.org/News/2015/20150817.
Debian.org/vote (s.f.). «Debian Voting Information». url: http://www.debian.org/vote/.
— (2014). «General Resolution: code of conduct». url: http://www.debian.org/vote/2014/vote_002.
— (2006). «General Resolution: Handling source-less firmware in the Linux kernel». url: http://www.debian.org/vote/2006/vote_007.
— (2008a). «General Resolution: Lenny and resolving DFSG violations». url: http://www.debian.org/vote/2008/vote_003.
— (2005). «General Resolution: Declassification of debian-private list archives». url: http://www.debian.org/vote/2005/vote_002.
— (2016). «General Resolution: Declassifying debian-private». url: http://www.debian.org/vote/2016/vote_004.
— (2007). «General Resolution: Endorse the concept of Debian Maintainers». url: http://www.debian.org/vote/2007/vote_003.
— (2010). «General Resolution: Debian project members». url: http://www.debian.org/vote/2010/vote_002.
— (2008b). «General Resolution: Project membership procedures». url: http://www.debian.org/vote/2008/vote_002.
— (2012). «General Resolution: Diversity statement». url: http://www.debian.org/vote/2012/vote_002.

Dep-team (s.f.[a]). «DEP - Debian Enhancement Proposals». url: http://dep-team.pages.debian.net/.

— (s.f.[b]). «Clarifying policies and workflows for Non Maintainer Uploads (NMUs)». url: http://dep-team.pages.debian.net/deps/dep1.html.

— (s.f.[c]). «Patch Tagging Guidelines». url: http://dep-team.pages.debian.net/deps/dep3/.

— (s.f.[d]). «Recommended layout for Git packaging repositories». url: http://dep-team.pages.debian.net/deps/dep14/.

— (s.f.[e]). «Machine-readable debian/copyright». url: http://dep-team.pages.debian.net/deps/dep5/.

Dsa.debian.org (s.f.). «Debian System Administrators». url: http://dsa.debian.org/.

Freexian.com (s.f.[a]). «Debian Long Term Support». url: http://www.freexian.com/services/debian-lts.html.

— (s.f.[b]). «Debian Long Term Support - Details». url: http://www.freexian.com/services/debian-lts-details.html.

Fsf.org (s.f.[a]). «Página principal de la *Free Software Foundation*». url: https://www.fsf.org/.

— (s.f.[b]). «What is free software and why is it so important for society?» url: http://www.fsf.org/about/what-is-free-software.

— (s.f.[c]). «The Principles of Community-Oriented GPL Enforcement». url: http://www.fsf.org/licensing/enforcement-principles.

Ftp-master.debian.org (s.f.[a]). «FTP Master server». url: http://ftp-master.debian.org.

— (s.f.[b]). «Debian NEW and BYHAND Packages». url: http://ftp-master.debian.org/new.html.

Geekfeminism.wikia.com (s.f.). «Debian-devel-games trolling». url: http://geekfeminism.wikia.com/wiki/Debian-devel-games_trolling.

Github.com (s.f.). «GitHub». url: http://github.com/.

GNOME (s.f.). «GNOME – Página principal». url: http://www.gnome.org/.

Gnu.org (s.f.[a]). «GNU – Página principal». url: http://www.gnu.org/.

— (s.f.[b]). «GNU General Public License». url: http://www.gnu.org/licenses/gpl-3.0.html.

— (s.f.[c]). «What is free software?» url: http://www.gnu.org/philosophy/free-sw.

Gnu.org (s.f.[d]). «Why “Free Software” is better than “Open Source”». url: http://www.gnu.org/philosophy/free-software-for-freedom.

Gnupg.org (s.f.[a]). «The GNU Privacy Guard – Página principal». url: http://gnupg.org/.

— (s.f.[b]). «Documentation». url: http://gnupg.org/documentation/index.html.

Ircphelp.org (s.f.). «Irchelp.org». url: http://www.irchelp.org/.

Kernel.org (s.f.). «The Linux Kernel Archives». url: http://www.kernel.org/.

Libresoft.es (s.f.). «Libresoft». url: http://libresoft.es/.

Linfo.org (s.f.[a]). «The Linux Information Project». url: http://www.linfo.org/.

— (s.f.[b]). «Software Definition». url: http://www.linfo.org/software.html.

— (s.f.[c]). «Source Code Definition». url: http://www.linfo.org/source_code.html.

Literateprogramming.com (s.f.). «Literate Programming – Página principal». url: http://www.literateprogramming.com/index.html.

Lkml.org (ago. de 2000). «Re: SCO: ”thread creation is about a thousand times faster than on native Linux”». url: http://lkml.org/lkml/2000/8/25/132.

Meetbot.debian.net (s.f.). «Index of /debian-ctte». url: http://meetbot.debian.net/debian-ctte/.

Meetings-archive.debian.net (s.f.[a]). «850 Meet the technical committee». url: http://meetings-archive.debian.net/pub/debian-meetings/2012/debconf12/low/850_Meet_the_Technical_Committee.ogv.

— (s.f.[b]). «926 Hijacking packages for fun and profit». url: http://meetings-archive.debian.net/pub/debian-meetings/2012/debconf12/archival/926_Hijacking_packages_for_fun_and_profit.ogv.

— (s.f.[c]). «Q&A session with Linus Torvalds». url: http://meetings-archive.debian.net/pub/debian-meetings/2014/debconf14/webm/QA_with_Linus_Torvalds.webm.

— (s.f.[d]). «Debian’s Central Role in the Future of Software Freedom». url: http://meetings-archive.debian.net/pub/debian-meetings/2015/debconf15/Debians_Central_Role_in_the_Future_of_Software_Freedom.webm.

Mentors.debian.net (s.f.[a]). «Mentors – Página principal». url: http://mentors.debian.net/.

— (s.f.[b]). «The sponsoring process». url: http://mentors.debian.net/sponsors.

Nm.debian.org (s.f.[a]). «Debian New Members». url: http://nm.debian.org/.

— (s.f.[b]). «Debian New Member - Statistics». url: http://nm.debian.org/public/stats.
— (s.f.[c]). «Debian New Member – Join the NM process». url: http://nm.debian.org/public/newnm.
Opensource.org (s.f.). «The Open Source Definition». url: http://opensource.org/docs/osd.
Outreachy.org (s.f.). «Outreachy – Página principal». url: http://www.outreachy.org/.
Packages.qa.debian.org (s.f.). «Debian Package Tracking System». url: http://packages.qa.debian.org/common/index.html.
Penta.debconf.org (s.f.[a]). «Hijacking packages for fun and profit». url: http://penta.debconf.org/dc12_schedule/events/926.en.html.
— (s.f.[b]). «DebConf9. I18n sessions 1/4». url: http://penta.debconf.org/dc9_schedule/events/400.en.html.
Perl-team (s.f.). «Debian Perl Group». url: http://perl-team.pages.debian.net/.
Pet.debian.net (s.f.). «Package Entropy Tracker». url: http://pet.debian.net/pkg-perl/pet.cgi.
Planet.debian.org (s.f.). «Planet Debian». url: http://planet.debian.org/.
Portfolio.debian.net (s.f.). «Debian Member Portfolio Service». url: http://portfolio.debian.net/.
Pypi.org (s.f.). «Python Package Index». url: http://pypi.org/.
Pypl.github.io (s.f.). «PYPL PopularitY of Programming Language». url: http://pypl.github.io/PYPL.html.
Python.org (s.f.). «Python.org – Página principal». url: http://www.python.org/.
Qa.debian.org (s.f.). «Debian Quality Assurance». url: http://qa.debian.org/.
Riseup.net (s.f.). «Encrypted Email with OpenPGP». url: http://riseup.net/es/security/message-security/openpgp.
Salsa.debian.org (s.f.[a]). «Salsa – Página principal». url: http://salsa.debian.org/public.
— (s.f.[b]). «Debian/tech-ctte». url: http://salsa.debian.org/debian/tech-ctte.
— (3 de nov. de 2013a). «Change default desktop to xfce.» url: http://salsa.debian.org/installer-team/tasksel/commit/dfca406eb694e0ac00ea04b12fc912237e01c9b5.
— (19 de sep. de 2014). «switch default desktop to Gnome». url: http://salsa.debian.org/installer-team/tasksel/commit/dce99f5f8d84e4c885e6beb4cc1bb5bb1d9ee6d7.

Salsa.debian.org (s.f.[c]). «nm-templates». url: http://salsa.debian.org/nm-team/nm-templates.

— (2 de sep. de 2013b). «Better descriptions for DCs, now that we have a name for them». url: http://salsa.debian.org/nm-team/nm.debian.org/commit/01b0666ed6d4381539cdb445ab688d1dda9c4774.

Sfconservancy.org (s.f.[a]). «About Software Freedom Conservancy, Inc.» url: http://sfconservancy.org/about/.

— (17 de ago. de 2015). «Conservancy and Debian announce Copyright Aggregation Project». url: http://sfconservancy.org/news/2015/aug/17/debian/.

— (s.f.[b]). «The Principles of Community-Oriented GPL Enforcement». url: http://sfconservancy.org/copyleft-compliance/principles.html.

SPI.org (s.f.). «SPI – Página principal». url: http://www.spi-inc.org/.

Timeline.debian.net (s.f.). «Timeline of the Debian Project». url: http://timeline.debian.net/.

Tiobe.com (s.f.). «TIOBE index». url: http://www.tiobe.com/tiobe-index//.

Tldp.org (s.f.). «Linux Loadable Kernel Module HOWTO». url: http://www.tldp.org/HOWTO/Module-HOWTO/.

Ubuntu.com (s.f.[a]). «Ubuntu – Página principal». url: http://www.ubuntu.com.

— (s.f.[b]). «Ubuntu Code of Conduct v2.0». url: http://www.ubuntu.com/community/code-of-conduct.

Udd.debian.org (s.f.). «Ultimate database». url: http://udd.debian.org/.

Wiki.debconf.org (s.f.[a]). «DebConf wiki – Página principal». url: http://wiki.debconf.org/.

— (s.f.[b]). «Teams. Chairs». url: http://wiki.debconf.org/wiki/Teams/Chairs.

Wiki.debian.org (s.f.[a]). «*Wiki* de Debian». url: http://wiki.debian.org.

— (s.f.[b]). «IRC». url: http://wiki.debian.org/IRC.

— (s.f.[c]). «IRC Netiquette». url: http://wiki.debian.org/IRC/Netiquette.

— (s.f.[d]). «Python». url: http://wiki.debian.org/Python.

— (s.f.[e]). «How to use BTS». url: http://wiki.debian.org/HowtoUseBTS.

— (s.f.[f]). «MeetBot». url: http://wiki.debian.org/MeetBot.

— (s.f.[g]). «Discussions After Lenny». url: http://wiki.debian.org/DiscussionsAfterLenny.

— (s.f.[h]). «Trusted Organization Criteria». url: http://wiki.debian.org/Teams/DPL/TrustedOrganizationCriteria.

— (s.f.[i]). «Declassification of the debian-private archives». url: http://wiki.debian.org/DebianPrivateDeclassification.

— (s.f.[j]). «Teams». url: http://wiki.debian.org/Teams.
— (s.f.[k]). «Debian Perl Group». url: http://wiki.debian.org/Teams/DebianPerlGroup.
— (s.f.[l]). «Non-maintainer upload». url: http://wiki.debian.org/NonMaintainerUpload.
— (s.f.[m]). «Low Threshold Nmu». url: http://wiki.debian.org/LowThresholdNmu.
— (s.f.[n]). «Debian Account Manager (DAM) resources». url: http://wiki.debian.org/DAManager.
— (s.f.[o]). «Requalification Jessie». url: http://wiki.debian.org/DebianDesktop/Requalification/Jessie.
— (s.f.[p]). «Debian Services». url: http://wiki.debian.org/Services.
— (s.f.[q]). «Why Debian For Developers». url: http://wiki.debian.org/WhyDebianForDevelopers.
— (s.f.[r]). «Binary Package». url: https://wiki.debian.org/Packaging/BinaryPackage.
— (s.f.[s]). «Source Package». url: http://wiki.debian.org/Packaging/SourcePackage.
— (s.f.[t]). «Packaging Intro». url: http://wiki.debian.org/Packaging/Intro.
— (s.f.[u]). «Using Quilt». url: http://wiki.debian.org/UsingQuilt.
— (s.f.[v]). «DebSrc3.0». url: http://wiki.debian.org/Projects/DebSrc3.0.
— (s.f.[w]). «Debian Package Management». url: http://wiki.debian.org/DebianPackageManagement.
— (s.f.[x]). «Advantages For Upstream». url: http://wiki.debian.org/AdvantagesForUpstream.
— (s.f.[y]). «Upstream Guide». url: http://wiki.debian.org/UpstreamGuide.
— (s.f.[z]). «Alioth Migration». url: http://wiki.debian.org/Salsa/AliothMigration.
— (s.f.[aa]). «Building A Package. Source In VCS». url: http://wiki.debian.org/BuildingAPackage/SourceInVCS.
— (s.f.[ab]). «Packaging With Git». url: http://wiki.debian.org/PackagingWithGit.
— (s.f.[ac]). «Fundraising». url: http://wiki.debian.org/Fundraising.
— (s.f.[ad]). «Debian Sponsoring Guidelines». url: http://wiki.debian.org/Teams/DPL/SponsoringGuidelines.

Wiki.debian.org (s.f.[ae]). «Budget Ideas». url: http://wiki.debian.org/BudgetIdeas.
— (s.f.[af]). «Organizations holding assets in trust for Debian». url: http://wiki.debian.org/Teams/Treasurer/Organizations.
— (s.f.[ag]). «Debian Long Term Support». url: http://wiki.debian.org/LTS.
— (s.f.[ah]). «Dunc-Tank external funding experiment». url: http://wiki.debian.org/PressCoverage2007#Dunc-Tank_external_funding_experiment.
— (s.f.[ai]). «New Queue». url: http://wiki.debian.org/NewQueue.
— (s.f.[aj]). «Summer of Code 2018». url: http://wiki.debian.org/SummerOfCode2018.
— (s.f.[ak]). «Planet Debian». url: http://wiki.debian.org/PlanetDebian.
— (s.f.[al]). «Sprints Howto». url: http://wiki.debian.org/Sprints/HowTo.
— (s.f.[am]). «Sprints». url: http://wiki.debian.org/Sprints.
— (s.f.[an]). «Glossary». url: http://wiki.debian.org/Glossary.
— (s.f.[ao]). «Debian Membership». url: http://wiki.debian.org/DebianMembership.
— (s.f.[ap]). «New Member Front Desk». url: http://wiki.debian.org/Teams/FrontDesk.
— (s.f.[aq]). «Maintainers». url: http://wiki.debian.org/Maintainers.
— (s.f.[ar]). «Debian Maintainer». url: http://wiki.debian.org/DebianMaintainer.
— (s.f.[as]). «Reformed Membership Process». url: http://wiki.debian.org/Projects/ReformedMembershipProcess.
— (s.f.[at]). «Debian Developer». url: http://wiki.debian.org/DebianDeveloper.
— (s.f.[au]). «Contributors Information Sources». url: http://wiki.debian.org/ContributorsInformationSources.
— (s.f.[av]). «MIA». url: http://wiki.debian.org/Teams/MIA.
— (s.f.[aw]). «Keyring Maint». url: http://wiki.debian.org/Teams/KeyringMaint.
— (s.f.[ax]). «Keysigning». url: http://wiki.debian.org/Keysigning.
Wikipedia (s.f.[a]). «Mao (card game)». url: http://en.wikipedia.org/wiki/Mao_(card_game).
— (s.f.[b]). «Language localisation». url: http://en.wikipedia.org/wiki/Language_localisation.
— (s.f.[c]). «Measuring programming language popularity». url: http://en.wikipedia.org/wiki/Measuring_programming_language_popularity.

— (s.f.[d]). «Benevolent dictator for life». url: http://en.wikipedia.org/wiki/Benevolent_dictator_for_life.
— (s.f.[e]). «Unix philosophy». url: http://en.wikipedia.org/wiki/Unix_philosophy.
— (s.f.[f]). «Proxy server». url: http://en.wikipedia.org/wiki/Proxy_server.
— (s.f.[g]). «Nyan Cat». url: http://en.wikipedia.org/wiki/Nyan_Cat.
Xfce (s.f.). «Xfce – Página principal». url: http://xfce.org/.

Apéndices

A Contrato social

Puede consultarse *online* en Debian.org (s.f.[ae]).

A.1 «Contrato social» con la comunidad de software libre

1. **Debian permanecerá 100% libre**

   «*Las directrices de software libre de Debian*» (DFSG) son el criterio que nosotros utilizamos para determinar si el software es «*libre*» o no. Prometemos mantener el sistema GNU/Linux así como todos sus componentes completamente libres de acuerdo con este criterio. No obstante, daremos soporte también a aquellos usuarios que desarrollen y ejecuten software no libre en Debian pero nunca haremos que el sistema tenga que utilizar obligatoriamente un componente que no sea libre.

2. **Contribuiremos a la comunidad de software libre**

   Cuando escribamos nuevos componentes del sistema Debian, los licenciaremos de forma consistente con nuestra definición de software libre. Haremos el mejor sistema que podamos, de forma que el software libre tenga amplia difusión y uso. Enviaremos parches, mejoras, peticiones de los usuarios, etc. a los autores originales (esto se conoce en inglés como «*upstream*», N. del T.) del software incluido en nuestro sistema.

3. **No ocultaremos los problemas**

   Mantendremos nuestra base de datos de informes de error accesible al público en todo momento. Los informes de error que los usuarios envíen serán visibles por el resto de usuarios de forma inmediata.

4. **Nuestra prioridad son nuestros usuarios y el software libre**

   Nos guiaremos por las necesidades de nuestros usuarios y de la comunidad del software libre. Sus intereses serán una prioridad para nosotros. Daremos soporte a las necesidades de nuestros usuarios para que puedan trabajar en muchos tipos distintos de entornos de trabajo. No pondremos objeciones al software no libre que vaya a ejecutarse sobre Debian ni cobraremos a las personas que quieran desarrollar o usar ese tipo de software (no libre). Permitiremos a otros crear distribuciones de valor añadido basadas en Debian sin cobrarles nada por ello. Es más, entregaremos un sistema integrado de alta calidad sin restricciones legales que pudieran prevenir este tipo de uso.

5. **Trabajos que no siguen nuestros estándares de software libre**

   Reconocemos que algunos de nuestros usuarios necesitan usar trabajos que no sigan las directrices de software libre de Debian (DFSG). Por ello, hemos creado las secciones «contrib» y «non-free» en nuestro archivo para estos trabajos. Los paquetes en estas secciones no son parte del sistema Debian, aunque han sido configurados para usarse con Debian. Animamos a los distribuidores de CDs a que lean las licencias de los paquetes en estas secciones para poder determinar si pueden distribuir este software en sus CDs. Así pues, aunque los trabajos que no sean libres no son parte de Debian, damos soporte para su uso, y proporcionamos infraestructura (como nuestro sistema de informe de errores y listas de distribución) para paquetes no libres.

B DFSG

Forman parte del Contrato social, y pueden consultarse *online* en Debian.org (s.f.[af]).

B.1 Las Directrices de Software Libre de Debian (DFSG)

1. **Libre redistribución**

   La licencia de un componente de Debian no puede restringir a un tercero el vender o entregar el programa como parte de una distribución mayor que contiene programas de diferentes fuentes. La licencia no debe solicitar «royalties» u otras comisiones para su venta.

2. **Código fuente**

   El programa debe incluir el código fuente completo, y debe permitir la distribución en forma de código fuente y en forma compilada (binario).

3. **Trabajos derivados**

   La licencia debe permitir modificaciones y trabajos derivados y debe permitir que estos se distribuyan bajo los mismos términos que la licencia del programa original.

4. **Integridad del código fuente del autor**

La licencia puede restringir la distribución del código fuente en forma modificada «**sólo**» si la licencia permite la distribución de «parches» («patch files») para poder modificar el código fuente original del programa en el momento de compilarlo. La licencia debe permitir explícitamente la distribución de software a partir de código fuente modificado. La licencia puede obligar a los trabajos derivados a llevar un nombre o número de versión diferentes del programa original. *Esto es un compromiso. El grupo de Debian anima a todos los autores a no restringir ningún fichero, fuente o compilado, de ser modificado.*

5. **No discriminación contra personas o grupos**

   La licencia no debe discriminar a ninguna persona o grupo de personas.

6. **No discriminación en función de la finalidad perseguida**

   La licencia no puede restringir el uso del programa para una finalidad determinada. Por ejemplo, no puede restringir el uso del programa a empresas con fines comerciales, o en investigación genética.

7. **Distribución de la licencia**

   Los derechos y libertades de uso asociados al programa deben aplicarse en la misma forma a todos aquellos a los que se redistribuya el programa, sin necesidad de pedir una licencia adicional para estas terceras partes.

8. **La licencia no ha de ser específica para Debian**

   Los derechos asociados al programa no deben depender de que el programa sea parte o no del sistema Debian. Si el programa es extraído de Debian y usado o distribuido sin Debian, pero manteniendo el resto de las condiciones de la licencia, todos aquellos a los que el programa se redistribuya deben tener los mismos derechos que los dados cuando forma parte de Debian.

9. **La licencia no debe contaminar a otros programas**

   La licencia no debe poner restricciones sobre otros programas que se distribuyan junto con el programa licenciado. Por ejemplo, la licencia no puede insistir que todos los demás programas distribuidos sobre el mismo medio deben ser software libre.

10. **Ejemplos de licencias**

    Las licencias «GPL», «BSD», y « Artística» son ejemplos de licencias que nosotros consideramos «libres».

C The Debian Manifesto

Puede consultarse *online* en Debian.org/doc (s.f.[i]).

Written by Ian A. Murdock, Revised 01/06/94

C.1 What is Debian Linux?

Debian Linux is a brand-new kind of Linux distribution. Rather than being developed by one isolated individual or group, as other distributions of Linux have been developed in the past, Debian is being developed openly in the spirit of Linux and GNU. The primary purpose of the Debian project is to finally create a distribution that lives up to the Linux name. Debian is being carefully and conscientiously put together and will be maintained and supported with similar care.

It is also an attempt to create a non-commercial distribution that will be able to effectively compete in the commercial market. It will eventually be distributed by The Free Software Foundation on CD-ROM, and The Debian Linux Association will offer the distribution on floppy disk and tape along with printed manuals, technical support and other end-user essentials. All of the above will be available at little more than cost, and the excess will be put toward further development of free software for all users. Such distribution is essential to the success of the Linux operating system in the commercial market, and it must be done by organizations in a position to successfully advance and advocate free software without

the pressure of profits or returns.

C.2 Why is Debian being constructed?

Distributions are essential to the future of Linux. Essentially, they eliminate the need for the user to locate, download, compile, install and integrate a fairly large number of essential tools to assemble a working Linux system. Instead, the burden of system construction is placed on the distribution creator, whose work can be shared with thousands of other users. Almost all users of Linux will get their first taste of it through a distribution, and most users will continue to use a distribution for the sake of convenience even after they are familiar with the operating system. Thus, distributions play a very important role indeed.

Despite their obvious importance, distributions have attracted little attention from developers. There is a simple reason for this: they are neither easy nor glamorous to construct and require a great deal of ongoing effort from the creator to keep the distribution bug-free and up-to-date. It is one thing to put together a system from scratch; it is quite another to ensure that the system is easy for others to install, is installable and usable under a wide variety of hardware configurations, contains software that others will find useful, and is updated when the components themselves are improved.

Many distributions have started out as fairly good systems, but as time passes attention to maintaining the distribution becomes a secondary concern. A case-in-point is the Softlanding Linux System (better known as SLS). It is quite possibly the most bug-ridden and badly maintained Linux distribution available; unfortunately, it is also quite possibly the most popular. It is, without question, the distribution that attracts the most attention from the many commercial «distributors» of Linux that have surfaced to capitalize on the growing popularity of the operating system.

This is a bad combination indeed, as most people who obtain Linux from these «distributors» receive a bug-ridden and badly maintained Linux distribution. As if this wasn't bad enough, these «distributors» have a disturbing tendency to misleadingly advertise non-functional or extremely unstable «features» of their

product. Combine this with the fact that the buyers will, of course, expect the product to live up to its advertisement and the fact that many may believe it to be a commercial operating system (there is also a tendency not to mention that Linux is free nor that it is distributed under the GNU General Public License). To top it all off, these «distributors» are actually making enough money from their effort to justify buying larger advertisements in more magazines; it is the classic example of unacceptable behavior being rewarded by those who simply do not know any better. Clearly something needs to be done to remedy the situation.

C.3 How will Debian attempt to put an end to these problems?

The Debian design process is open to ensure that the system is of the highest quality and that it reflects the needs of the user community. By involving others with a wide range of abilities and backgrounds, Debian is able to be developed in a modular fashion. Its components are of high quality because those with expertise in a certain area are given the opportunity to construct or maintain the individual components of Debian involving that area. Involving others also ensures that valuable suggestions for improvement can be incorporated into the distribution during its development; thus, a distribution is created based on the needs and wants of the users rather than the needs and wants of the constructor. It is very difficult for one individual or small group to anticipate these needs and wants in advance without direct input from others.

Debian Linux will also be distributed on physical media by the Free Software Foundation and the Debian Linux Association. This provides Debian to users without access to the Internet or FTP and additionally makes products and services such as printed manuals and technical support available to all users of the system. In this way, Debian may be used by many more individuals and organizations than is otherwise possible, the focus will be on providing a first-class product and not on profits or returns, and the margin from the products and services provided may be used to improve the software itself for all users whether they paid to obtain it or not.

The Free Software Foundation plays an extremely important role in the future of Debian. By the simple fact that they will be distributing it, a message is sent to the world that Linux is not a commercial product and that it never should be, but that this does not mean that Linux will never be able to compete commercially. For those of you who disagree, I challenge you to rationalize the success of GNU Emacs and GCC, which are not commercial software but which have had quite an impact on the commercial market regardless of that fact.

The time has come to concentrate on the future of Linux rather than on the destructive goal of enriching oneself at the expense of the entire Linux community and its future. The development and distribution of Debian may not be the answer to the problems that I have outlined in the Manifesto, but I hope that it will at least attract enough attention to these problems to allow them to be solved.