Investigaciones Europeas de Dirección y Economía de la Empresa
Vol. 14, Nº 3, 2008, pp. 79-90, ISSN: 1135-2523

ESTUDIO Y ANÁLISIS DE LAS LISTAS DE DISTRIBUCIÓN EN PROYECTOS DE SOFTWARE DE CÓDIGO ABIERTO COMO MEDIO PARA COMPARTIR CONOCIMIENTO

Martínez Torres, M.R.
Toral Marín, S.L.
Barrero García, F.J.
Universidad de Sevilla

Recibido: 30 de abril de 2007
Aceptado: 1 de septiembre de 2008

RESUMEN: Este trabajo explora el papel desempeñado por las listas de distribución en proyectos de software de código abierto como herramienta para compartir conocimiento y resolver problemas. Uno de los principales beneficios que las empresas pueden obtener del uso de software de código abierto es la colaboración informal en desarrollo de aplicaciones. La herramienta más importante para esa colaboración y coordinación son las listas de correo, seguido por los foros de discusión asíncronos, informes de bugs y chat. Uno de los argumentos principales a la hora de decidirse por una distribución de Linux embebido es el soporte proporcionado a los desarrolladores. Pero generalmente, es difícil tomar una decisión a priori sin conocer si el soporte proporcionado será lo suficientemente bueno durante todo el futuro desarrollo del proyecto. Para ayudar en esta tarea, este trabajo se centra en analizar el comportamiento y la actividad de las listas de correo para extraer una serie de parámetros que puedan aportar información sobre la calidad y la evolución de la lista. Esta información resultaría relevante para analizar y decidir sobre la mejor distribución de Linux embebido a utilizar.
PALABRAS CLAVE: Software de Código Abierto, Listas de Distribución, Linux embebido, Análisis Factor, Sistemas de Información.

STUDY AND ANALYSIS OF OPEN SOURCE SOFTWARE PROJECTS MAILING LISTS AS A TOOL FOR KNOWLEDGE SHARING

ABSTRACT: This research explores the role of mailing lists in open source software projects as a tool for knowledge sharing and problem resolution. One of the benefits that firms can derive from using Open Source Software (OSS) is informal development collaboration. The primary tool for collaboration and coordination are group mailing lists, followed by asynchronous discussion forums, bug reports, and chat. One of the main arguments when deciding about an embedded Linux distribution is the support provided to developers. However, it is usually difficult to decide a priori if the provided support will be good enough for the future development of the project. Particularly, the behaviour and activity of mailing list are analyzed to extract a set of parameters that could inform about the quality and the evolution of the list. This information could be useful to decide the best embedded distribution to be implemented.
KEYWORDS: Open Source Software, Mailing Lists, Embedded Linux, Factor Analysis, Information Systems

1. INTRODUCCIÓN

El software de código abierto es un modelo de desarrollo del software en el que el código fuente se encuentra disponible para que los programadoras puedan verlo, leerlo, modificarlo y redistribuirlo sin las restricciones de derechos de propiedad típicos del software propietario distribuido bajo licencia (Waring y Maddocks, 2005). La innovación constante se consigue gracias a las contribuciones individuales de desarrolladores ampliamente distribuidos geográficamente. Según su definición, software de código abierto no supone únicamente el acceso al código fuente, sino una serie de reglas sobre la redistribución, trabajos derivados, integridad del código fuente del autor y no discriminación (Perens, 1997).

La investigación en el ámbito del software de código abierto se ha ido incrementando a lo largo de los años, analizando áreas tales como la motivación de programadores y desarrolladores (Bonaccorsi, A., & Rossi, 2003; Hertel *et al.*, 2003), los beneficios proporcionados por el software de código abierto (Kogut & Metiu, 2001; Spinellis & Szyperski, 2004; Spinellis, 2006), las implicaciones para el sector público (Waring & Maddocks, 2005; Applewhite, 2003; McDonald *et al*., 2003) o las licencias de dominio público (Gambardella & May, 2006; Valimaki & Oksanen, 2005). No obstante, la investigación sobre el papel desempeñado por las listas de correo como fuente de soporte para el software de código abierto es más limitada, aunque se trate de una característica común a todos los proyectos de desarrollo de software de este tipo. Sowe *et al.* (2006) identificaron un grupo dentro de los participantes en proyectos de software de código abierto, los "brokers" del conocimiento, usando para ello las listas de distribución que sirven de soporte. Algunos otros estudios explican este fenómeno desde el punto de vista de las comunidades de práctica (Kogut & Metiu, 2001; Li et al., 2006). Esta será la perspectiva a seguir en el desarrollo del presente trabajo al analizar las listas de distribución.

Existen muchos paquetes informáticos que se desarrollan como comunidades de código abierto, pero de entre todos ellos Linux es quizás el ejemplo más claro. Linux es un sistema operativo UNIX desarrollado por Linus Torvalds y una comunidad de programadores de Internet en 1991. Su modelo de desarrollo en código abierto y su licencia general pública GNU (GPL, General Public License), bajo la cual se puso en circulación, permitió que el código fuente del kernel de Linux estuviese disponible gratuitamente para uso personal o comercial. Fue precisamente la disponibilidad del código fuente la que estimuló a los programadores a contribuir al desarrollo del kernel de Linux. El resultado final ha sido un desarrollo distribuido geográficamente, en el que los programadores trabajan en ubicaciones arbitrarias, con escasos o ningún encuentro de tipo presencial, y que coordinan sus actividades casi exclusivamente por medios electrónicos (Mockus et al., 2002). La existencia de Linux demuestra que los procesos de los proyectos de software de código abierto pueden producir un software de alta calidad aun cuando no empleen los mecanismos tradicionales de coordinación tales como planes, planificaciones, niveles de diseño y definición de procesos. Los defensores de esta forma de hacer software argumentan que los defectos se encuentran y solucionan muy rápidamente, ya que hay "muchos ojos" observando posibles problemas y aportando soluciones ("Ley de Linus"). El código se escribe con mayor cuidado y creatividad, ya que los desarrolladores trabajan en aquello por lo que sienten una auténtica pasión (Raymond, 1999)

El objetivo del presente trabajo es analizar el desarrollo colaborativo en un campo más concreto dentro del software de código abierto, que es el de Linux embebido. Un sistema embebido es un dispositivo que lleva un procesador dentro, aunque el usuario final no sea necesariamente consciente de que existe. Ejemplos de sistemas embebidos se encuentran permanentemente a nuestro alrededor, desde dispositivos de electrónica de consumo (PDA, teléfonos móviles, reproductores MP3, cámaras digitales, …), pasando por la electrónica de vehículos de automoción y sistemas de seguridad y vigilancia, hasta equipos de comunicación e instrumentación o equipos aeroespaciales (Abbott, 2003). El Linux embebido muestra algunas diferencias significativas con respecto al Linux tradicional para PC. En primer lugar, los sistemas embebidos incorporan una variedad más amplia de dispositivos de entrada y salida que los PCs. Eso supone que los programadores de sistemas embebidos tienen que trabajar frecuentemente con el hardware del sistema. En segundo lugar, y en contraste con otros proyectos de software de código abierto, la mayoría de las contribuciones en este campo no

viene de voluntarios o aficionados sino de firmas comerciales, muchas de las cuales son empresas dedicadas a Linux embebido. Por último, el hecho de que Linux se distribuya con licencia GPL hace que las empresas consideren revelar sus propios desarrollos software, a diferencia del caso de usar un software propietario. Todas estas características fomentan el trabajo colaborativo y justifican la necesidad de analizar las posibilidades de soporte mediante medios electrónicos.

El presente trabajo se organiza de la siguiente forma: la sección II está dedicada a introducir las comunidades de práctica en proyectos de software de código abierto, con unos patrones de desarrollo radicalmente diferentes del software propietario. La sección III presenta el caso de estudio basado en un procesador con Linux embebido. La metodología utilizada para procesar los datos recogidos de la lista de distribución se explica en la sección IV, y los resultados obtenidos a cerca de los parámetros que describen el comportamiento y la evolución de las listas de distribución se detallan en la sección V. Estos resultados son discutidos junto con las conclusiones en la sección VI.

2. COMUNIDADES DE PRÁCTICA EN PROYECTOS DE SOFTWARE DE CÓDIGO ABIERTO

El éxito de proyectos de desarrollo de alta complejidad tecnológica depende fuertemente de la habilidad de los miembros del equipo para interaccionar de manera productiva, de modo que el conocimiento relevante pueda ser adquirido, generado y transmitido de modo efectivo en coste y en tiempo. En este sentido, el concepto de comunidades de práctica ha ido ganando atención como medio para explicar como el conocimiento y el aprendizaje se desarrollan en contextos sociales específicos (Garrety et al., 2004).

Las comunidades de práctica (CoP) se organizan en torno a un conjunto de actividades limitada y sus miembros se encuentran generalmente en contacto directo unos con otros. Ellos desarrollan sus propias rutinas, reglas formales e informales y, como resultado del aprendizaje, las prácticas desarrolladas evolucionan (Wenger, 1998).

La noción de CoP sugiere que los límites de las comunidades no se corresponden con los típicos límites funcionales. En su lugar, incluye redes basadas en prácticas y en personas. Los miembros de las comunidades típicamente emplean su tiempo ayudándose unos a otros a resolver problemas. La importancia de las comunidades de práctica ha sido reconocida en numerosos trabajos previos (Brown & Duguid, 1998; Boland & Tenkasi, 1995; Pan & Leidner, 2003; Zárraga & Saá, 2005).

El éxito de los proyectos de software de código abierto, como Linux, Apache, Sendmail o Jabber, puede explicarse desde esta perspectiva. Estos paquetes, pública y gratuitamente disponibles, han alcanzado una amplia difusión así como una calidad comparable e incluso superior a otros productos comerciales similares, a pesar de que no estuvieran soportados por ninguna compañía comercial, al menos en sus inicios. Estos proyectos fueron creciendo a través de comunidades de desarrolladores geográficamente dispersos y colaborando por internet (Gruber & Henkel, 2006).

Los programadores que trabajan en proyectos de software de código abierto colaboran sin recibir ningua compensación monetaria por ello, al menos directa. Entre las motivaciones para realizar esa colaboración se encuentra la necesidad de nuevas funcionalidades en el software, apoyo a otros desarrollos, reputación entre colegas, diversión, necesidad de nuevos

aprendizajes, búsqueda de nuevos horizontes laborales, el deseo de devolver el apoyo recibido de la comunidad de programadores o una aversión al software propietario, en general.

Un caso particularmente interesante e importante dentro del software de código abierto es el representado por el llamado Linux embebido. Este término se refiere a las variantes del sistema operativo Linux adaptadas a los dispositivos embebidos. Los sistemas embebidos se encuentran permanentemente a nuestro alrededor y Linux se ha convertido en la elección más habitual como sistema operativo interno. No obstante, la heterogeneidad de los sistemas embebidos fuerza habitualmente a adaptar el sistema operativo a los dispositivos específicos en los que se implemente. Muchas veces, los drivers específicos requeridos en determinadas aplicaciones no se encuentran disponibles o bien no soportan la funcionalidad completa de ese periférico. En consecuencia, es necesario desarrollar un importante trabajo de adaptación del sistema operativo.

Típicamente, son firmas comerciales las que desarrollan Linux embebido, fundamentalmente fabricantes de bienes o productos electrónicos. Para una compañía puede resultar interesante participar en una comunidad que dé soporte a un proyecto de software de código abierto o fomentar el desarrollo de una comunidad en torno a su propio software (Henkel, 2003). Las ventajes e inconvenientes del desarrollo de software de código abierto con fines comerciales ha sido estudiado por numerosos autores (Henkel, 2006: Dahlander & Magnusson, 2005; Johnson, 2006). Básicamente, se pueden distinguir cuatro categorías de potenciales beneficios: establecimiento de un estándar mejorando la compatibilidad, incremento de la demanda de bienes y servicios complementarios, aprovechamiento del soporte de desarrollos externos realizados por la comunidad y divulgación de la excelencia técnica. En el caso de Linux embebido, el beneficio del soporte de desarrollos externos se puede obtener por dos vías diferentes:

- Como librerías de código fuente u objeto, o como módulos que resuelven alguna funcionalidad critica del sistema embebido. Por ejemplo, todos los protocolos y componentes de la pila TCP/IP proporcionan servicios muy útiles si nuestro sistema va a estar conectado en red. La pila TCP/IP se encuentra incluida por defecto en Linux embebido, pero sería de pago en el caso de utilizar un sistema operativo propietario.
- Como soporte proporcionado por las listas de distribución, bastante útil en el caso de sistemas embebidos que son muy heterogéneos y que requieren un trabajo de adaptación del software muy superior al de los PCs. Desde un punto de vista estructural, estas listas de distribución funcionan a modo de repositorios en los que cada programador o desarrollador puede suscribirse, facilitando a todos los participantes la búsqueda de expertos en un área determinada (Gutwin et al., 2004). Los participantes en la lista pueden enviar libremente preguntas o cuestiones y obtener respuestas a los problemas planteados.

3. CASO DE ESTUDIO: LINUX EMBEBIDO PARA ARM

De los 6.2 billones de procesadores fabricados en 2002, menos del 2% fueron a parar a PCs, Macs o estaciones de trabajo Unix. El otro 98% formó parte de sistemas embebidos (Ganssle, 2003) en aplicaciones que van desde juguetes y semáforos hasta equipos de consumo o aplicaciones domóticas o satelitales.

Al comienzo de su desarrollo, los sistemas embebidos no incorporaban sistemas operativos. El desarrollo del software se realizaba actuando directamente sobre el hardware del sistema, sin incluir prácticamente capacidad de procesamiento multitarea ni de acceso en red. A medida que los sistemas embebidos fueron aumentando su complejidad y su capacidad de procesamiento, también fueron en incremento los requerimientos sobre ellos: capacidad de procesamiento multitarea, gestión de memoria y de procesos, comunicación entre tareas y procesos, acceso a redes de comunicación, etc. Un sistema embebido puede, por ejemplo, incluir un servidor web que permita la configuración remota de un equipo mediante una página accesible por Internet, así como accesos remotos de mantenimiento y actualización (Raghavan et al., 2006; Yaghmour, 2003). Todos estos requerimientos obligaron el uso de sistemas operativos en los sistemas embebidos. Aunque actualmente existen varios sistemas operativos embebidos disponibles (Wind River's VxWorks, Microsoft Windows CE, QNX Neutrino, etc), Linux es claramente el líder en este tipo de sistemas. Las principales ventajas de Linux embebido frente a otros sistemas operativos propietario son independencia del proveedor, plazo de comercialización y bajo coste. No obstante, hay que tener en cuenta que la licencia pública general (GPL), bajo la que Linux se distribuye, estipula que la distribución del software original o de versiones modificadas debe realizarse bajo las mismas condiciones definidas por GPL. En particular, el comprador o cliente tiene derecho a recibir el código fuente así como los derechos de modificar y redistribuir el software (Henkel, 2003; Free Software Foundation: GNU General Public License, 2006). De todos modos, GPL no obliga a hacer público el software modificado ni excluye poder vender el software (eso sí, sin royalties por unidad vendida).

Debido a la gran heterogeneidad de los sistemas embebidos, no existe una versión estándar de Linux, sino que hay que hablar de muchas distribuciones de Linux que cubren una o varias arquitecturas procesadoras. Todas ellas tienen en común los mismos componentes básicos, incluyendo el kernel de Linux, drivers, comandos, entornos de ventanas, utilidades básicas y librerías. Las diferencias entre las distribuciones se centran normalmente en cuál de los cientos de utilidades existentes se incluyen, en los módulos añadidos (tanto en código fuente como propietario), en las modificaciones del kernel y en la gestión de los procesos de instalación, configuración, mantenimiento y actualización. La Tabla 1 proporciona la ubicación más apropiada de los diferentes kernel de Linux para diferentes arquitecturas procesadoras.

Tabla 1. Ubicación de los kernel de Linux para diferentes arquitecturas procesadoras.

Arquitectura procesadora	Ubicación del kernel
x86	http://www.kernel.org/
ARM	http://www.arm.linux.org.uk/developer/
PowerPC	http://penguinppc.org/
MIPS	http://www.linux-mips.org/
SuperH	http://linuxsh.sourceforge.net/
M68k	http://www.linux-m68k.org/

Fuente: Elaboración propia

ARM (Advanced RISC Machine) es una familia de procesadores mantenida y promovida por ARM Holdings Ltd. A diferencia de otros fabricantes de procesadores como IBM, Motorota o Intel, ARM Holdings no fabrica sus propios procesadores. En su lugar, diseña los núcleos de la CPU para sus clientes a los que les carga un canon de licencia por el diseño, permitiéndoles a cambio fabricar sus chips basados en este núcleo ARM. Actualmente, Intel, Toshiba, Samsung, Freescale y otros muchos fabrican chips con núcleos ARM. Su arquitectura es tremendamente popular y existen cientos de proveedores con productos y servicios basados en él. Hoy en día, Linux soporta más de 1200 tarjetas basadas en microprocesadores con un ARM.

Uno de los foros más efectivos con mayor calidad para encontrar respuesta a los problemas sobre Linux en un ARM es la lista de distribución lists.arm.linux.org.uk. Este servicio es gratuito para la comunidad Linux y es frecuentemente consultado por muchos desarrolladores de Linux embebido sobre ARM. El uso de las listas de distribución como fuente de soporte resulta de gran interés, ya que es una forma sencilla y útil de conseguir adaptar una distribución de Linux a las características particulares de un proyecto concreto. Como consecuencia, la caracterización de las listas de distribución es una cuestión de interés para obtener información a cerca de las posibilidades de soporte y la tendencia de una distribución de Linux y del procesador subyacente. Este es el objetivo primordial de este trabajo, en el que se analizará la lista de distribución sobre Linux para ARM con idea de deducir los parámetros que indiquen es éxito de esa distribución y su evolución en el tiempo.

4. METODOLOGÍA

Las listas de distribución permiten la interacción ente los buscadores y los proveedores de conocimiento y entre los desarrolladores y usuarios del software, jugando un papel muy importante tanto en su difusión como en su mejora. Este trabajo pretende profundizar en el conocimiento de estas actividades e interacciones, y en su importancia para el éxito de los proyectos de software de código abierto. La cuestión central del trabajo es la caracterización de las lista de distribución, utilizando como caso de estudio la lista sobre ARM ubicada en lists.arm.linux.org.uk.

Hasta ahora, la investigación desarrollada en torno a esta cuestión ha sido más bien escasa. La mayoría de los trabajos se concentran en la naturaleza de la participación en comunidad en las listas de distribución asociadas a proyectos de software de código abierto (Mockus et al., 2003; Krogh et l., 2003; Lakhani & Hippel, 2003). Con la excepción de algún estudio (Sowe et al., 2006), poca ha sido la atención prestada a la actividad de las listas de distribución en este tipo de proyectos. En este trabajo no se pretende únicamente identificar un conjunto de parámetros capaces de explicar la actividad de las listas de distribución, sino también analizar su evolución en el tiempo. La hipótesis fundamental es que la evolución de las listas de distribución proporciona información sobre las posibilidades de conseguir un buen soporte técnico durante el desarrollo de un sistema embebido, y éste es un parámetro fundamental a la hora de elegir una distribución de Linux embebida.

Los datos de las listas de distribución en proyectos de código abierto se encuentran disponibles y son fáciles de obtener, lo que permite medir toda una serie de parámetros sobre su actividad. Nuestro estudio ha utilizado los datos de la lista de distribución de Linux para ARM entre los años 2001 y 2006, midiéndose los parámetros detallados en la Tabla 2. Dado que se pretende analizar la evolución en el tiempo, estos parámetros han sido medidos por meses.

Tabla 2. Parámetros medidos en la lista de distribución.

Variable	Descripción
Var01	Número de mensajes
Var02	Número de autores
Var03	Número de mensajes por autor
Var04	Número de temas diferentes
Var05	Número de temas sin respuesta
Var06	Número medio de mensajes con el mismo tema
Var07	Desviación típica de mensajes con el mismo tema
Var08	Máximo número de mensajes con el mismo tema
Var09	Número medio de mensajes enviados por el mismo autor
Var10	Desviación típica de los mensajes enviados por el mismo autor
Var11	Máximo número de mensajes enviados por el mismo autor
Var12	Número medio de mensajes enviados por el mismo autor a temas diferentes
Var13	Desviación típica de mensajes enviados por el mismo autor a temas diferentes

Fuente: Elaboración propia

Los factores latentes que explican la actividad de la lista de distribución se extraerán haciendo uso del análisis factor. De acuerdo a esta técnica estadística, las variables medidas depende de un conjunto más pequeño de factores latentes no observables. Debido a que cada factor puede afectar a varias variables, se les suele denominar también factores comunes. Cada variable puede expresarse como una combinación lineal de los factores comunes o latentes, con unos coeficientes denominados cargas. El análisis factor resulta de gran utilidad para identificar relaciones entre variables que no son directamente observables, segmentando una muestra en grupos relativamente homogéneos.

Las cargas estimadas de un análisis factor sin rotación suelen tener una estructura complicada. El objetivo de la rotación factor ortogonal es precisamente encontrar una parametrización en la que cada variable solamente tenga un número reducido de cargas elevadas, o, lo que es lo mismo, que sólo se vea afectada por un pequeño número de factores. El análisis factor con rotación permite que los factores representen constructos unidimensionales, más fácilmente interpretables.

En nuestro estudio, la muestra está formada por los meses entre Enero de 2001 y Diciembre de 2006 (72 casos en total). Los factores latentes que se obtengan se utilizarán para realizar un estudio longitudinal. Los casos de la muestra (meses) serán agrupados conforme a los factores latentes resultantes. Los grupos que se obtengan mostrarán si los patrones mensuales son repetitivos, o si por el contrario existen tendencias claras temporales en la evolución de los factores.

5. RESULTADOS

Las principales aplicaciones del análisis factor consisten en reducir el número de variables y en detectar la estructura de relaciones entre variables, permitiendo su clasificación. El análisis factor trata de identificar los factores subyacentes que explican el patrón de correlaciones dentro de un conjunto de variables observadas. Los autovalores de la matriz de covarianzas de la muestra se muestran en la Tabla 3. Habitualmente, el análisis factor toma un número de factores igual al número de autovalores mayor que la unidad. En nuestro caso, son tres los autovalores que satisfacen esta condición. La Tabla 3 también muestra el porcentaje de la varianza explicada por cada factor así como la varianza explicada acumulada. Los tres

factores anteriores explican un 88,2 % del total de la varianza de la muestra, ofreciendo por tanto una adecuada representación de las trece variables de partida.

Tabla 3. Varianza total explicada.

Factor	Autovalores		
	Valor	% de Varianza	% Varianza acumulada
1	7,270	55,920	55,920
2	2,728	20,986	76,906
3	1,468	11,294	88,200
4	,734	5,644	93,843
5	,362	2,786	96,629
6	,200	1,538	98,167
7	,107	,826	98,993
8	,064	,492	99,485
9	,033	,254	99,739
10	,024	,187	99,926
11	,006	,046	99,972
12	,004	,028	100,000
13	3,20E-005	,000	100,000

Fuente: Elaboración propia

Las cargas de los factores se estiman a partir de los autovectores asociados. Normalmente, suele resultar complicado realizar una correcta interpretación de los factores a partir de las cargas estimadas. Afortunadamente, las cargas de los factores se pueden rotar multiplicándolos por una matriz ortogonal. Las cargas rotadas preservan las propiedades de las originales y además facilitan la interpretación de los factores.

El método Varimax es un método de rotación ortogonal que minimiza el número de variables con una carga elevada en cada factor. Este método simplifica la interpretación de los factores. La Tabla 4 muestra las cargas de los factores rotadas mediante el método Varimax.

Tabla 4. Cargas de los factores rotadas por el método Varimx.

	Cargas de los factores		
	1	2	3
V1	,488	,835	,139
V2	,128	,931	,047
V3	,860	,340	,248
V4	,345	,931	-,072
V5	,100	,912	-,022
V6	,450	-,150	,622
V7	,089	-,116	,947
V8	,102	,209	,897
V9	,863	,331	,248
V10	,933	,123	,224
V11	,873	,157	,183
V12	,627	,704	-,106
V13	,852	,420	-,122

Para extraer el significado de cada factor, nos movemos horizontalmente de izquierda a derecha observando las cargas de cada una de las trece variables, e identificando para qué factor alcanza su valor máximo. Las cargas de los factores se consideran significativas a partir del umbral de 0,6 (Rencher, 2002). El resultado es una serie de variables asociadas a cada factor que conducen a los siguientes factores latentes:

- Factor 1 (variables V3, V9, V10, V11 y V13): el primer grupo representa aquellos casos en los que existe un ratio óptimo de mensajes por autor. Los autores son bastante activos y el número de mensajes sin respuesta es bajo, por lo que existen un debate efectivo alrededor de las cuestiones propuestas. El valor intermedio de la carga de V6 y el bajo valor de la carga de V7 significan que el número de mensajes en cada tema tiene un valor adecuado con una desviación típica baja, lo que garantiza un debate en profundidad. Podría decirse que una lista de distribución que se encuentre en esta situación goza de una buena salud.
- Factor 2 (variables V1, V2, V4, V5 y V12): el segundo de lo grupos se caracteriza fundamentalmente por el alto número de mensajes sin respuesta. Son muchos los mensajes enviados por los autores, pero poco el debate generado. El bajo valor de la carga en V6, acompañado del bajo valor de V7, sugiere que hay pocos mensajes alrededor de las cuestiones enviadas. Una lista de distribución en estas condiciones resulta inútil, ya que no proporciona un soporte a las cuestiones enviadas.
- Factor 3 (variables V6, V7 y V8): el último grupo representa el caso de una lista de distribución con pocos autores pero muy activos en las cuestiones suscitadas. Aunque esta situación no tiene por qué ser negativa, puede ocurrir que el debate se encuentre excesivamente polarizado en torno a algunos temas polémicos. En cualquier caso, no debería ser una situación prolongada en el tiempo debido a la falta de variedad en los temas tratados en la lista.

Además de la carga de los factores, el análisis factor también proporciona los valores de los factores. Estos valores pueden utilizarse para asignar cada uno de los casos de la muestra (meses) a alguno o ninguno de los factores latentes obtenidos. La asignación se ha realizado al factor para el que ese caso tiene un valor máximo, siempre y cuando dicho valor sea mayor de 0,1 (Rencher, 2002). De las 72 observaciones de la muestra, 58 han sido clasificadas en alguno de los tres factores latentes obtenidos. La Tabla 5 muestra el valor medio de los trece parámetros de la lista de distribución considerados, medidos sobre los casos asociados a cada factor.

Tabla 5. Valor medio de los parámetros de la lista de distribución para los casos asociados a cada factor.

Factor	N	V1	V2	V3	V4	V5	V6	V7
1	19	28,34	21,76	40,11	28,13	26,34	32,11	27,26
2	21	35,79	42,83	23,19	39,24	39,57	20,62	16,33
3	18	23,39	22,11	25,67	19,58	21,08	37,11	47,22
Total	58							
Factor	**N**	**V8**	**V9**	**V10**	**V11**	**V12**	**V13**	
1	19	22,34	40,11	42,89	41,21	34,63	41,74	
2	21	24,21	23,00	19,62	21,48	33,38	27,81	
3	18	43,22	25,89	26,89	26,50	19,56	18,56	
Total	58							

Fuente: Elaboración propia

Los resultados obtenidos muestran claramente cómo las variables asignadas al primer factor (V3, V9, V10, V11 y V13) tienen un valor medio claramente superior al valor medio de estas mismas variables en el resto de factores. Lo mismo podría decirse para los otros dos factores. Para validar estos resultados, la hipótesis nula de igualdad de medias en las poblaciones debería ser rechazada. Para ello, se ha aplicado el test de Kruscal-Wallis. Los resultados obtenidos aparecen detallados en la Tabla 6. La hipótesis nula puede rechazarse en todos los casos excepto para la variable 1, el número total de mensajes, donde el nivel de significación está por encima de 0,05. Esto significa que el número de mensajes no es un parámetro discriminatorio entre los factores y que por tanto no debe uilizarse para modelar el comportamiento de las listas de distribución.

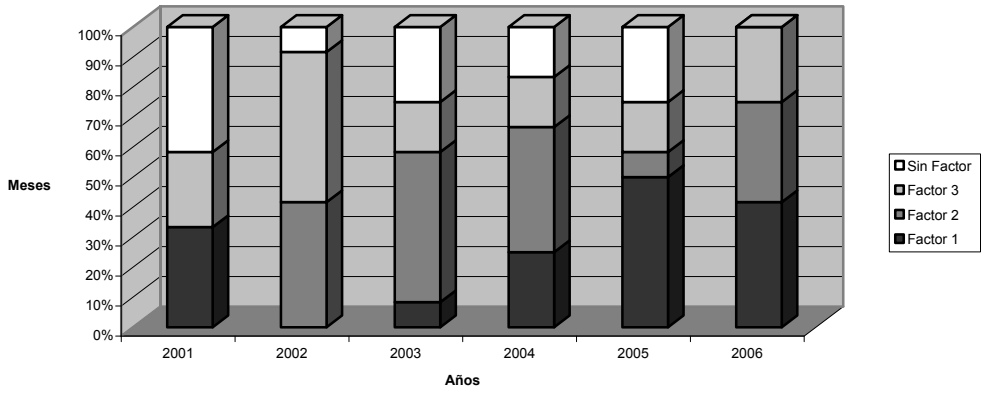
Tabla 6. Test de Kruscal-Wallis.

	V1	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**
Chi-Square	5,357	20,551	11,353	13,335	12,650	9,917	32,925
df	2	2	2	2	2	2	2
Asymp. Sig.	,069	,000	,003	,001	,002	,007	,000
	V8	**V9**	**V10**	**V11**	**V12**	**V13**	
Chi-Square	17,629	11,428	19,574	14,503	9,107	17,748	
df	2	2	2	2	2	2	
Asymp. Sig.	,000	,003	,000	,001	,011	,000	

Fuente: Elaboración propia

Las observaciones de nuestro caso de estudio representan los 72 meses que van de Enero de 2001 a Diciembre de 2006. 58 de las 72 observaciones han sido clasificadas en uno de los tres factores latentes obtenidos. Estas categorizaciones muestran realmente una evolución temporal, ilustrada en la Figura 1.

Figura 1. Evolución temporal de la lista de distribución de Linux sobre ARM.



Los meses que no han sido asignados a ningún factor se muestran en color blanco mientras que los meses asignados a algunos de los factores se muestran en distintas tonalidades de gris. El primer año, 2001, no se puede considerar significativo dado que son muchos los meses que no han sido asignados a ningún factor. Durante 2002, el comportamiento dominante es el representado por los factores 3 y 2. Obsérvese que durante esta etapa los desarrolladores comienzan a realizar aplicaciones basadas en Linux sobre plataformas ARM. La lista de distribución no es todavía una buena fuente de soporte: a veces el debate se polariza demasiado en torno a pocas cuestiones (factor 3) y otras veces no resulta útil a los desarrolladores (factor 2). Los años 2003 y 2004 corresponden a una etapa de crecimiento en la que las aplicaciones basadas en Linux sobre ARM se consolidan y la lista de distribución tiene un comportamiento más positivo. La suma de meses con el comportamiento de los factores 1 y 3 crece en 2003 y durante 2004 alcanza el valor de los meses con un comportamiento negativo como el descrito por el factor 2. Los dos años siguientes corresponden ya a una etapa consolidada, donde los desarrollos con Linux embebido sobre ARM son habituales. El número de meses con el comportamiento descrito por los factores 1 y 3 permanece constante y mayor que el número de meses con un comportamiento negativo.

De los resultados anteriores se observa una evolución concordante con la evolución del mercado en lo referente a aplicaciones basadas en Linux sobre ARM. El soporte proporcionado por la lista de distribución también contribuye a la evolución exitosa de esta familia de procesadores.

6. CONCLUSIONES

El propósito de este trabajo ha sido la caracterización de una lista de distribución como fuente de soporte en proyectos de software de código abierto. Básicamente se han obtenido tres tipos de comportamiento en el caso de la lista de distribución de Linux sobre ARM. El primer comportamiento identificado representa una verdadera comunidad de práctica con autores activos y variedad de temas tratados. La lista de distribución se puede considerar en este caso una buena fuente de soporte para los desarrolladores. El segundo tipo de comportamiento es bastante negativo, ya que se caracteriza por muchas cuestiones no contestadas. Este hecho causa un efecto de frustración entre los desarrolladores, al no encontrar un soporte adecuado. El útimo comportamiento se caracteriza por un debate hiper-concentrado en pocos temas. Aunque este tipo de debate no tiene por qué ser negativo, una situación prolongada de este tipo puede originar una falta de variedad de temas y, nuevamente, frustración entre los desarrolladores.

Para poder realizar un estudio longitudinal en el tiempo, la muestra se ha organizado por meses. Básicamente, se ha identificado la presencia de alguno de los tres comportamientos anteriores en los últimos seis años. Esta evolución muestra la tendencia de la lista de distribución como fuente de soporte y, al mismo tiempo, la evolución del procesador subyacente. En el caso de la lista de distribución de Linux sobre ARM, se puede decir que la lista goza de muy buena salud, al igual que la evolución en el mercado de los procesadores ARM.

La principal limitación del trabajo es la generalización de los resultados obtenidos. Se trata de un estudio exploratorio que debería ser replicado utilizando otras listas de distribución de Linux, pero sobre procesadores diferentes.

BIBLIOGRAFÍA

ABBOTT, D. (2003): *Linux for Embedded and Real Time Applications*, Newnes, Elsevier Science, USA, 2003.
APPLEWHITE, A. (2003): "Should governments go Open Source?" *IEEE Software*, Vol. 20, nº 4, pp. 88–91.
BOLAND, R., TENKASI, R. (1995): "Perspective making and perspective taking in communities of knowing", *Organization Science*, Vol. 6, nº 4, pp. 350–372.
BONACCORSI, A., & ROSSI, C. (2003): "Why Open Source Software can succeed", *Research Policy*, Vol. 32, pp. 1243–1258.

BROWN, J.S., DUGUID, P. (1998): "Organizing knowledge", *California Management Review Spring,* Vol. 40, nº 3, pp. 90–106.
DAHLANDER, L., MAGNUSSON, M.G. (2005): "Relationships between open source software companies and communities: Observations from Nordic firms", *Research Policy*, Vol. 34, nº 4, pp. 481–493.
FREE SOFTWARE FOUNDATION: GNU GENERAL PUBLIC LICENSE (2001): http://www.fsf.org/licenses/gpl.html, [on line] con acceso 2006-12-14.
GAMBARDELLA, A., MAY, B.H. (2006): "Proprietary versus public domain licensing of software and research products", *Research Policy*, Vol. 35, pp. 875–892.
GANSSLE, JACK and MICHAEL BARR. (2003): *Embedded Systems Dictionary*, Lawrence, KS: CMP Books.
GARRETY, K., ROBERTSON, P.L., BADHAM, R. (2004): "Integrating communities of practice in technology development projects", *International Journal of Project Management*, Vol. 22, pp. 351–358.
GRUBER, M., HENKEL, J. (2006): "New ventures based on open innovation – an empirical analysis of start-up firms in embedded Linux", *International Journal of Technology Management*, Vol. 33, nº 4, pp. 356 – 372.
GUTWIN, C., PENNER, R., SCHNEIDER K. (2004): "Group awareness in distributed software development", *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, Chicago, pp. 72–81.
HENKEL, J. (2003): "Software development in embedded Linux - informal collaboration of competing firms", *Proceedings der 6. Internationalen Tagung Wirtschaftsinformatik*, Dresden, pp. 1-20.
HENKEL, J. (2006): "Selective revealing in open innovation processes: The case of embedded Linux", *Research Policy*, Vol. 35, pp. 953–969.
HERTEL, G., NIEDNER, S., & HERRMANN, S. (2003): "Motivation of software developers in Open Source projects: An Internet-based survey of contributors to the Linux kernel", *Research Policy*, Vol. 32, pp. 1159–1177.
JOHNSON, J.P. (2006): "Collaboration, peer review and open source software", *Information Economics and Policy*, Vol. 18, pp. 477–497.
KOGUT, B., & METIU, A. (2001): "Open Source Software and distributed innovation", *Oxford Review of Economic Policy*, Vol. 17, nº 2, pp. 248–264.
KROGH, G., SPAETH, S., LAKHANI, K.. (2003): "Community, joining, and specialisation in open source software innovation: a case study", *Research Policy*, Vol. 32, pp. 1217–1241.
LAKHANI, K., HIPPEL, E. (2003): "How open source software works: 'free' user to user assistance", *Research Policy*, Vol. 32, pp. 923–943.
LI, X., MONTAZEMI, A.R., YUAN, Y. (2006): "Agent-based buddy-finding methodology for knowledge sharing", *Information & Management*, Vol. 43, pp. 283–296
MCDONALD, C. J., SCHADOW, G., BARNES, M., DEXTER, P., OVERHAGE, J. M., MAMLIN, B. (2003): "Open Source Software in medical informatics—why, how and what", *International Journal of Medical Informatics*, Vol. 69, nº 2/3, pp. 175–184.
MOCKUS, A., FIELDING, R.T., HERBSLEB, J.D. (2002): "Two Case Studies of Open Source Software Development: Apache and Mozilla", *ACM Transactions on Software Engineering and Methodology*, Vol. 11, nº 3, pp. 309–346.
PAN, S.L., LEIDNER, D.E. (2003): "Bridging communities of practice with information technology in pursuit of global knowledge sharing", *Journal of Strategic Information Systems*, Vol. 12, pp. 71–88.
PERENS, B. (1997): *The Open Source definition*, http://www.opensource.org/docs/definition.php, [on line] con acceso 17/12/2006.
RAGHAVAN, P., LAD, A., NEELAKANDAN, S. (2006): *Embedded Linux System Design and Development,* Auerbach Publications, Taylor and Francis Group.
RAYMOND, E. S. (1999): *The cathedral and the bazaar*. Available at http://www.tuxedo.org/□esr/writings/cathedral-bazaar/, [on line] con acceso 17/12/2006.
RENCHER, A.C. (2002): *Methods of Multivariate Analysis*, 2nd ed. Wiley Series in Probability and Statistics. John Wiley & Sons.
SOWE, S., STAMELOS, I., ANGELIS,L.(2006): "Identifying knowledge brokers that yield software engineering knowledge in OSS projects", *Information and Software Technology*, Vol. 48, pp. 1025–1033.
SPINELLIS, D., & SZYPERSKI, C. (2004): "How is Open Source affecting software development?", *IEEE Software*, Vol. 21, nº 1, pp. 28–33.
SPINELLIS, D. (2006): "Open Source and Professional Advancement", *IEEE Software*, Vol. 23, nº 5, pp. 70–71.
VALIMAKI, M, OKSANEN, V. (2005): "The impact of free and open source licensing on operating system software markets", *Telematics and Informatics*, Vol. 22, pp. 97–11.
WARING, T., MADDOCKS, P. (2005): "Open Source Software implementation in the UKpublic sector: Evidence from the field and implications for the future", *International Journal of Information Management*, Vol. 25, pp. 411-428.
WENGER E. (1998): *Communities of practice: learning, meaning, and identity*, Cambridge: Cambridge University Press; 1998.
YAGHMOUR, K. (2003): *Building Embedded Linux Systems.* O'Reilly.
ZÁRRAGA OBERTY, C. & SAÁ PÉREZ, P. (2005): "Comunidades de práctica: equipos de trabajo para la gestión del conocimiento", *Revista Europea de Dirección y Economía de la Empresa*, Vol. 14, nº 2, pp. 145-158.