Journal of Universal Computer Science, vol. 15, no. 3 (2009), 648-664
submitted: 26/2/08, accepted: 29/1/09, appeared: 1/2/09 © J.UCS

Modelling Mailing List Behaviour in Open Source Projects: the Case of ARM Embedded Linux

Sergio L. Toral
(University of Seville, Spain
toral@esi.us.es)

Rocío Martínez Torres
(University of Seville, Spain
rmtorres@us.es)

Federico Barrero
(University of Seville, Spain
fbarrero@esi.us.es)

Abstract: One of the benefits firms can derive from using Open Source Software (OSS) is informal development collaboration, and the primary tool for collaboration and coordination are group mailing lists. The purpose of the paper is modelling mailing lists behaviour in OSS projects, using a set of descriptors that could inform about their quality and their evolution. As a case study, a mailing list focused on ARM embedded Linux has been selected. Messages posted to this list from 2001 to 2006 have been extracted, and factor analysis has been applied to obtain the underlying patterns of behaviours. Theory about communities of practice has been used to understand the meaning of the extracted patterns. Their time distribution is finally described. The paper provides new insights into the behaviour of mailing list as a source of support for OSS projects and highlights the importance of an involved core of individuals inside the community.

Keywords: virtual communities, web-based communities, collaborative work, information systems, social networks, open source projects, embedded systems
Categories: H.3.5, H.4.3

1 Introduction

Open Source Software (OSS) is a model of computer software development where the source code is available for programmers to view, read, modify and re-distribute without the property right restrictions of proprietary software [Waring and Maddocks, 2005]. Constant innovation is achieved thanks to individual contributions of geographically widely distributed developers. In accordance with its definition, open source doesn't just mean access to the source code, but also some rules about redistribution, derived works, integrity of the author's source code, and no discrimination [Perens, 1997].

Research in the area of OSS has become more extensive in recent years, and it has examined areas such as motivation of programmers and developers [Bonaccorsi and Rossi, 2003; Hertel et al., 2003], the benefits of OSS [Kogut and Metiu, 2001; Spinellis and Szyperski, 2004; Spinellis, 2006], and its implications for the public

sector [Waring and Maddocks, 2005; Applewhite, 2003; McDonald et al., 2003], public domain licensing [Gambardella and May, 2006; Valimaki and Oksanen, 2005], or successful OSS development [Perkins, 1999; Dinh-Trong and Bieman, 2005]. However, research about the role of mailing lists as a source of support for OSS is more limited, although it is a common feature of OSS development projects. Sowe et al. [2006] identified one group of OSS project participants—knowledge brokers—using open source projects mailing lists. Some other studies explain this phenomenon from the perspective of communities of practice [Kogut and Metiu, 2001; Li et al., 2006]. This perspective will be followed in the paper when analyzing the role of mailing list.

There are many software programs designed in open-source communities, but Linux is perhaps the clearest example. Linux is an operating system developed by Linus Torvalds and a community of programmers across the Internet in 1991. Its open source development model and GNU General Public License (GPL), under which Linux is released, allowed all the Linux kernel source code to be freely available for personal or commercial use. The availability of the source code encouraged developers to contribute to the Linux kernel. The result is a geographically distributed development, where developers work in arbitrary locations, rarely or never meet face to face, and coordinate their activity almost exclusively by means of email and bulletin boards [Mockus et al., 2002]. The existence of Linux demonstrates that OSS processes can produce high-quality and widely deployed software, although they lack many of the traditional mechanisms used to coordinate software development, such as plans, system-level design, schedules, and defined processes. It is claimed, for example, that defects are found and fixed very quickly because there are "many eyeballs" looking for the problems ("Linus's Law"). Code is written with more care and creativity, because developers are working only on things for which they have a real passion [Raymond, 1999].

This paper analyzes the behaviour of collaborative development in a somewhat unusual field of open source software, namely, embedded Linux. An embedded system is a device that has a computer inside it, but the user of the device does not necessarily know, or care, that the computer is there. Examples of embedded systems can be found everywhere: consumer electronic devices (PDA, cell phones, MP3 player, digital cameras, …), car engine control and security systems, communication and instrumentation equipments, aerospace applications, etc [Abbott, 2003].

Embedded Linux shows significant differences from conventional desktop Linux. First, embedded systems incorporate a wider variety of input/output devices than typical desktop computers. That means that embedded systems programmer often has to deal directly with the hardware. Second, in contrast to other typical open source projects, most contributions in this field do not come from volunteers or hobbyists, but from commercial firms, many of which are dedicated embedded Linux firms [Henkel, 2003]. Third, the fact that it comes under an OSS license like GPL makes firms consider revealing their own developments, as opposite to the case of proprietary software. These features encourage the collaborative development and justify the necessity of studying the behaviour of the networked support.

This paper is organized as follows. Section 2 gives some background about communities of practice in OSS development as opposite to proprietary software. Section 3 introduces the case study, giving some background about embedded Linux.

Section 4 describes the methodology used for processing the collected mailing list data. Section 5 shows the obtained results about mailing list behaviour and its time evolution. Finally, the empirical findings are interpreted in section 6.

2 Communities of Practice in OSS Projects

The success of complex technology development projects depends heavily on the ability of team members to interact productively so that relevant knowledge can be acquired, generated and circulated in a timely and cost-effective fashion. In this sense, the concept of communities of practice (CoP) has recently gained attention as a way of explaining how knowledge and learning develop in specific social contexts [Garrety et al., 2004; Droschl, 2004].

CoPs are organized around circumscribed sets of activities and their members are generally in direct contact with each other. They develop their own routines, formal and informal ‘‘rules’’, and practices evolve as a result of learning [Wenger, 1998]. The notion of CoP suggests that organizational community boundaries do not correspond with typical functional boundaries. Instead, it includes practice and person based networks. Members of CoP typically spend time helping each other to solve problems. The importance of CoP has been acknowledged in a number of previous works [Brown and Duguid, 1998; Boland and Tenkasi, 1995; Nowak & Wurst, 2003].

The success of OSS projects such as Linux, Apache, Sendmail, or Jabber can be explained from this perspective. These publicly and freely available software packages have reached wide diffusion, as well as a similar quality or even superior to that of commercial substitutes, despite the fact that they were not supported by any commercial company, at least in the earliest phases of their history. Rather, they grew out of geographically dispersed virtual communities of developers collaborating over the internet [Gruber and Henkel, 2006; Kolbitsch & Maurer, 2006]. Programmers working on public open source projects contribute their programming effort without receiving direct monetary compensation. Different sources of motivation have been identified: need for new functionality in the software, development support by others, reputation among peers, fun of programming, learning, signalling to the job market, the wish to give back to the OSS community, and an aversion to proprietary software in general. Recent empirical studies support that strong drivers for OSS programmers can be found in intrinsic motives such as the fun of programming, e.g. Hertel et al. [2003], Lakhani and Wolf [2005], and the gift culture surrounding OSS, e.g. Raymond [1999], Zeitlyn [2003], and extrinsic motives like software customization to their own needs [Niedner et al., 2000; Lakhani and Wolf, 2005]. The interrelationships between the motivations, participation, and performance of OSS developers have been analyzed by Roberts et al. [2006]

A particularly interesting and important instance of such OSS projects is “embedded Linux”. This term denotes variants of the Linux operating system that are adapted to embedded devices. Embedded devices are becoming more and more ubiquitous, and Linux has become one of the top choices as an operating system for them. Nevertheless, the heterogeneity of embedded devices forces to adapt the operating system to the specific device. Sometimes, drivers for specific peripherals are not available or they do not support the whole functionality of these peripherals.

So a huge work of customization of the open source operating system must be performed.

Typically, embedded Linux is developed by commercial firms, above all, manufacturers of goods containing embedded software. For a commercial firm, it may make sense to participate in a (community based) open source project, or to instigate a development community around its own software [Henkel, 2003]. Costs and benefits of commercial OSS development have been explored by various authors [Henkel, 2006; Dahlander and Magnusson, 2005; Johnson, 2006]. Basically, four categories of potential benefits can be distinguished according to these authors: setting a standard and enabling compatibility; increasing demand for complementary goods and services; benefiting from external development support, in particular from the OSS community; and signalling technical excellence or good OSS citizenship. In the case of embedded Linux, the benefit of external development support is obtained from two different sources:

- As object code libraries or modules that solve some critical functionality of the embedded systems. For instance, the network protocol stacks and components provide useful system services to an embedded application in a networked environment. The TCP/IP stack is included by default in embedded Linux, but it should be paid for an embedded proprietary operating system.
- As the support provided from mailing list, since embedded systems are highly heterogeneous and thus require more software adaptations than standard computers. Structurally, lists act as proxies for every programmer or developer subscribed to them [Gutwin et al., 2004], making it easy for project participants to find out who the experts are in a given area. Participants can subscribe to and freely post questions and get replies.

The analysis of underlying mailing lists will be performed attending to several typical features of collaboration in OSS projects.

The first one is the size of the community which can be measured through the number of messages, authors and threads of discussion (typically, OSS mailing lists can be sorted attending to each of these criteria). Some authors claim that the number of developers and contributions constitute a measure of the community success [Preece, 2001].

The second one is the activity of the mailing list. Typically, mailing lists are organized through threads of discussion. Threads are groups of messages sharing the same subject. A thread is initiated by someone who posts a message asking for help, suggesting some improvements, or just considering some new idea. Then people start answering this initial message, posting possible solutions, sources of information or just extending posted considerations. Some members of the community become engaged in a process of conceptualization, leading to some collective innovation and new knowledge. The result is a list of related messages where the sequence of reflections is detailed, so newcomers can follow expert reasoning step by step. Consequently, threads can be considered as the mechanism through which interactions take place [Jones et al., 2004]. The simplest way to classify threads is using their length, i.e. the total number of posts they contain. Posts per thread – how densely packed posts are in a collection of threads – turns out to be a reliable metric to

determine the degree of activity [Bonacci, 2004]. Several statistics related to the threads length will be considered in the proposed analysis.

Finally, a typical characteristic of virtual communities in general and OSS communities in particular is the fact that much of the OSS development is realized by a small percentage of individuals despite there are tens of thousands of developers available [Madey et al. 2004]. Such concentration constitutes participation inequality. Participation inequality is a well-known problem in physical collocated groups. It underlines a social and psychological phenomenon that distribution of participation is not random but hierarchical, with a few highly verbose members dominating the discussion [Kuk 2000]. This situation is frequently observed in OSS projects where the support and behaviour of mailing lists are strongly influenced by a hard core of participants. Some authors distinguish three groups in OSS communities [Mockus et al., 2002; Xu, Gao, Christley, & Madey, 2005]:

- Core members. They are responsible for guiding and coordinating the development of an OSS project. They are usually involved with the project during a long period of time and have made significant contributions to the development and evolution of the system [Madanmohan and Navelkar, 2004]. Moderators and leaders are included in this group.
- Active developers. They regularly make contributions to the project.
- Peripheral developers. They occasionally contribute with new features to the existing system. This contribution is irregular, and the period of involvement is short and sporadic. Free riders (people who just are seeking answers without making any contributions) are also included in this group.

The number of messages posted by the same author will be used as an indicator of the core group of developers while the number of messages posted by the same author to different threads will be used as a measure of the participation inequality.

3 Case Study: ARM Embedded Linux

Of the 6.2 billion processors manufactured in 2002, less than 2% became the brains of new PCs, Macs, and Unix workstations. The other 98% went into embedded systems [Ganssle and Barr, 2003]. From toys to traffic lights, from consumer equipments to household appliances or satellites and flight control, embedded systems are widely used [Barrero et al., 2008].

In the early days of embedded systems, no operating systems were used. There was in-house development of all the software that directly drived the hardware with almost no or very minimal multitasking and user interaction. But as more complex embedded systems started emerging, a growing list of features to be supported was also required: multitasking/multithreading, process and memory management, interprocess communication, etc. Increasingly, embedded system designers were called upon to include networking capabilities in their products. An embedded system might, for example, include a web server to enable web-based configuration. It might also enable remote login for maintenance and upgrading purposes [Raghavan et al., 2006; Yaghmour, 2003]. All of these requirements mandated the use of an operating system in embedded systems. Although a multitude of embedded operating systems are currently available (Wind River's VxWorks, Microsoft Windows CE, QNX

Neutrino, etc), Linux is firmly in first place as the operating system of choice for smart gadgets and embedded systems (Figure 1). Some of the advantages of embedded Linux against proprietary embedded operating systems are vendor independence, time to market and low cost. Nevertheless, the General Public License (GPL), under which Linux stands, stipulates that if someone distributes the software or modified versions of it, then this must be done under the conditions of the GPL. In particular, the recipient of the software must obtain the source code as well as the rights to modify and redistribute the software [Henkel, 2003; Free Software Foundation: GNU General Public License, 2001]. However, the GPL does not require making modified software publicly available (companies are free to make modifications and use them privately, without ever releasing them). Further, it does not exclude selling the software (for a one-time price, not per-unit royalties).

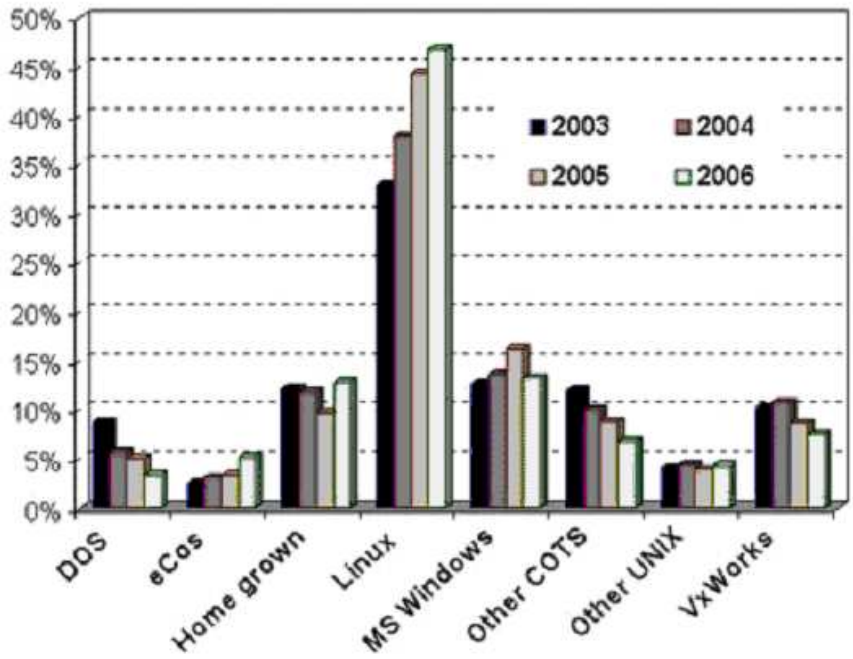

Figure 1: Embedded operating systems evolution (source: Linuxdevices.com, Snapshot of the embedded Linux market -- May, 2006)

Due to high heterogeneity of embedded devices, there is no standard version of embedded Linux. As a consequence, there are lots of Linux distributions covering one or several processor architectures. The distributions are variations of the same theme. All of them have in common the same basic components, including the Linux kernel, drivers, command shells (command processors), GUI and windowing frameworks, utilities and libraries. The distinctions between distributions are usually centered around which of the many hundreds of Linux utilities are included, what modules or

utilities (both open-source and proprietary) are added, kernel patches and modifications, and how the installation, configuration, maintenance and upgrade process is managed. Table I shows a list of sites where the most appropriate kernel for different processor architectures could be found.

Processor architecture	Most appropriate kernel location
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/

Table I: Most appropriate kernel sites for each processor architecture.

The ARM processor architecture, which stands for Advanced RISC Machine, is a family of processors maintained and promoted by ARM Holdings Ltd. Contrary to other chip manufacturers such as IBM, Motorola, and Intel, ARM Holdings does not manufacture its own processors. Instead, ARM designs the CPU cores for its customers based on the ARM core, charges customers licensing fees on the design, and lets them manufacture the chip wherever they see fit. Currently, ARM CPUs are manufactured by Intel, Toshiba, Samsung, Freescale, Texas Instruments, and many others. The ARM architecture is very popular in many fields of application and there are hundreds of vendors providing products and services around it. Today, Linux supports more than 1200 related ARM based boards.

One of the highest quality and most effective forum for finding answers to problems when working with ARM embedded Linux is the “lists.arm.linux.org.uk” mailing list [Yaghmour, 2003], although some other GNU/Linux distributions for the ARM architecture can be found (Debian, Metrowerks). This list server is provided for members of the Linux Community free of charge, and it is frequently consulted by many ARM embedded Linux developers. The use of mailing lists as a source of support should never be ignored, as it is the best way of adapting a Linux distribution to the particular features of a particular project. As a consequence, the characterization of mailing list is an important issue to obtain some information about the possibilities of support and the trend of a particular Linux distribution or the underlying processor. This is the main goal of this paper, in which the ARM mailing list will be analyzed to deduce the main indicators able to suggest the degree of success of a Linux distribution.

4 Methodology

Mailing lists support the interaction between knowledge seekers and knowledge providers, and between the developers and users of the software, playing a major role in the diffusion and improvement of the software. The aim of this paper is to develop a better understanding of these activities and interactions, and their importance for the success of OSS projects. The central question of this paper is the characterization of mailing lists, using the mailing list located at "lists.arm.linux.org.uk" as a case study
Little rigorous research has been conducted in this area yet. Most of them are focused on the nature of community participation in OSS projects mailing lists [Sowe et al., 2006; Krogh et al., 2003; Lakhani and Hippel, 2003]. With the exception of some studies [Sowe et al., 2006], less attention has been paid to mailing list activities in OSS projects. In this study, we are not just interested in identifying the set of descriptors which can explain the activity of the mailing list, but also in analyzing their evolution over time. We posit that the evolution of the mailing list provides information about the possibilities of achieving a good technical support when developing an embedded system and this is an essential criterion to be considered when choosing an embedded Linux distribution.

Open source projects mailing list data is widely available and easy to extract, providing an excellent infrastructure to study community participation in an OSS project. Data from ARM Linux mailing list located at "lists.arm.linux.org.uk" during the years 2001 to 2006 have been used in our research. A set of descriptors, illustrated in Table II, has been measured. As we are interested in time evolution, descriptors have been measured per month.

A total of 12010 messages posted by 2086 authors have been considered. Basically, mailing list archives can be sorted by messages, authors and threads of discussion. The three first descriptors from Table II are precisely their accounted values. When accounting authors, it is necessary to consider the fact that they are identified using aliases. Aliases usually correspond to a unique e-mail address, but this is not always truth. Header of messages should be processed to check there is no duplicity of aliases or e-mails [Bird et al., 2006]. Depth of discussion can be measured through the next three descriptors: the number of threads with just one message (that is, without an answer), the average number of messages in a thread, and their standard deviation. Standard deviation is used to distinguish between situations in which the debate is concentrated on a few threads from situations in which the debate is equally distributed in all the threads. Author activity is finally considered on the rest of descriptors: average and standard deviation of messages per author are the basic statistics with regard to author activity while average and standard deviation of messages posted by the same author to different threads are a measure of participation inequality.

Descriptor	Description
D1	No. of messages
D2	No. of authors
D3	No. of different threads of discussion
D4	No. of threads without an answer
D5	Average number of messages per threads
D6	Standard deviation of messages in threads
D7	Maximum number of messages in threads
D8	Average number of messages posted by the same author
D9	Standard deviation of messages posted by the same author
D10	Average number of messages posted by the same author to different threads
D11	Standard deviation of messages posted by the same author to different threads

Table II: Mailing list descriptors.

The latent factors which are able to explain the mailing list activity will be extracted using the Factor Analysis [Gorsuch, 1974]. Factor Analysis is a way to fit a model to multivariate data, estimating their interdependence. In the factor analysis model, the measured variables depend on a smaller number of unobserved (latent) factors. Because each factor may affect several variables in common, they are known as "common factors". Each variable is assumed to be dependent on a linear combination of the common factors, and the coefficients are known as loadings. Factor analysis is useful in identifying inter-relationships between variables which are not directly observable, segmenting a sample into relatively homogeneous segments [Aguila-Obra and Padilla-Meléndez, 2006; Martínez-Torres and Toral, in press].

The estimated loadings from an unrotated factor analysis fit can usually have a complicated structure. The goal of orthogonal factor rotation is to find a parameterization in which each variable has only a small number of large loadings, i.e., is affected by a small number of factors. The rotated factor analysis fit ensures that factors represent unidimensional constructs.

Our sample is made up of months, from January 2001 to December 2006 (72 cases). The obtained latent factors will be used to perform the longitudinal study. The cases of the sample (months) will be grouped according to the resulting factors. The resulting groups will show if the monthly patterns are repetitive through the years, or if there is a clear trend in the evolution of the latent factors.

5 Results

The main applications of factor analysis techniques are to classify and to reduce the number of variables, structuring the relationships between them. Factor analysis attempts to identify underlying variables or factors, which can explain the pattern of correlations within a set of observed variables [Dyba, 2005]. A factor analysis has been performed using the principal component method. The eigenvalues of the

sample covariance matrix, the percentage of variance explained by each factor and the cumulative variance are shown in Table III. In factor analysis it is usual to consider a number of factors equal to the number of eigenvalues higher than 1 and able to explain a high percentage of variance [Little and Gibson, 2003]. The results from Table III show that the first and second eigenvalues are higher than 1. Nevertheless, the third eigenvalue is very close to 1 and improve the cumulative explained variance up to 90%. That is the reason why a total of three factors have been considered in the presented analysis.

Factors	Initial Eigenvalues		
	Total	% of Variance	Cumulative %
1	5.937	53.977	53.977
2	2.524	22.942	76.919
3	1.298	11.796	88.715
4	.690	6.272	94.987
5	.199	1.809	96.797
6	.165	1.496	98.293
7	.092	.834	99.127
8	.059	.535	99.662
9	.027	.249	99.911
10	.006	.057	99.968
11	.004	.032	100.000

Table III: Total variance explained.

Descriptors	Factor Loadings		
	1	2	3
D1	.826	.501	.143
D2	.922	.145	.044
D3	.931	.346	-.066
D4	.928	.065	-.015
D5	-.178	.404	.905
D6	-.108	.056	.953
D7	.222	.060	.638
D8	.329	.859	.264
D9	.142	.898	.252
D10	.694	.651	-.097
D11	.431	.842	-.097

Table IV: Rotated factor loadings with varimax rotation.

Using the associated eigenvectors, factor loadings can be estimated. Sometimes, it is difficult to perform the right interpretation of factors using the estimated loadings. Fortunately, factor loading can be rotated through the multiplication by an orthogonal matrix. The rotated loadings preserve the essential properties of the original loadings.

Varimax method is an orthogonal rotation method that minimizes the number of variables that have high loadings on each factor. This method simplifies the interpretation of the factors. Table IV reports the rotated factor loadings with varimax rotation. To extract the meaning of each factor, we move horizontally, from left to right, across the three estimated loadings of each variable, identifying the highest loading and the corresponding factor. To assess significance of factor loadings, a threshold value of 0,35 was considered [Rencher, 2002]. The resulting aggregation of variables is highlighted in Table IV and leads to the following latent factors:

- Factor 1 (descriptors D1, D2, D3, and D4): the first group is characterized by the high value of accounted messages, authors and threads. But descriptor D4 also shows a high value in the number of threads without an answer. That means a lot of messages are posted by a lot of authors, but there is no debate. The low value of D5 and D6 loadings means that the average number of messages per thread is very low with a low standard deviation. Although the mailing list exhibits a high number of messages, authors and threads, there are no effective interactions among authors, so the list is not providing a good support to developers.
- Factor 2 (descriptors D7, D8, D9, and D10): the second group represents those cases in which there is an optimum ratio of messages per author. Authors are active, and the number of messages without an answer is low. The high standard deviation of messages posted by the same author to different threads illustrates the typical participation inequality in OSS projects. There is an effective debate around the proposed questions. The intermediate value of D5 loading and the low value of D6 loading imply an appropriate mean value of messages per thread, and a low standard deviation. Constantly, the arising questions are deeply debated. It can be said that a mailing list with the behavior described in this factor enjoys good health.
- Factor 3 (descriptors D5, D6): the last group is the case of a mailing list with few authors but very active around a few threads. The high value of D5 and D6 loadings means very long threads of discussion. It is the group with the smallest number of authors and threads (lowest value of D2 and D3 loadings). This situation suggests a hyper-debate around very few questions supported by very few authors, usually the core of the community. A hyper-debate has not to be considered necessarily bad, as sometimes the debate may be polarized around some polemic questions. Anyway, it should not be a situation extended in time due to the lack of variety of threads.

Once the patterns of behaviour have been obtained and identified through the three obtained factors, it would be interesting to assign one of these patterns to each observation (month) in order to perform a longitudinal study of the mailing list evolution. Factor scores will be used for this purpose. Factor scores are defined as estimates of the underlying factor values for each observation. Notice that factors represent the patterns of behaviour while observations are the 72 moths included in the analysis. Using factor scores, each of the observations (months) can be assigned to

one or none of the latent factors. The assignation has been performed using the maximum factor score, provided that this maximum value is higher than 0,1 [Rencher, 2002]. From the 72 observations included in the sample, 60 have been classified in one of the three latent factors obtained. Particularly, 22 observations (months) are classified in factor 1, 19 are associated to factor 2, and 19 to factor 3. The resting 12 observations could no be classified in any of the obtained factors. The association of an observation to a particular factor means that the exhibited behaviour during that month is primarily described by the descriptors associated to that factor. Table V shows the mean value of the ten mailing list descriptors using previous classification of observations.

Factor	N	D1	D2	D3	D4	D5	D6
1	22	200.39	86.23	80.16	31.30	2.49	2.04
2	19	147.13	68.47	68.47	23.55	2.78	2.30
3	19	123.58	66.32	58.03	23.79	2.75	3.03
Total	60						
Factor	N	D7	D8	D9	D10	D11	
1	22	12.00	2.33	2.61	1.73	1.69	
2	19	11.44	2.77	3.92	1.77	2.27	
3	19	19.11	2.37	3.05	1.52	1.40	
Total	60						

Table V: Mean values of the mailing list descriptors for the cases belonging to each latent factor.

The results show that the descriptors belonging to the first latent factor (D1, D2, D3, and D4) exhibit a significant higher mean value respect to the mean value in the other two latent factors. The same can be said for the two other factors. To validate these results, the null hypothesis of equal population means should be rejected. In general, the purpose of analysis of variance (ANOVA) is to test for significant differences between means. But ANOVA is based on the assumption that the sampling distribution is normal. This requirement can not be verified for the current sample because the sample is not large enough. Hopefully, non parametric tests do not rely on this assumption. Basically, there is at least one nonparametric equivalent for each parametric general type of test. The equivalent for ANOVA is the Kruskal Wallis test. The Kruskal Wallis test has been applied to check the null hypothesis. Its results are detailed in Table VI. As usual, statistically significance results are indicated by significance values below 0,05. The obtained results corroborate that and all descriptors are discriminant of the three patterns of behavior previously explained.

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11
Chi-Square	9,4	19.2	12.2	12.1	10.6	33.9	24.4	12.4	16.5	11.4	15.1
df	2	2	2	2	2	2	2	2	2	2	2
Asymp. Sig.	.009	.000	.002	.002	.005	.000	.000	.002	.000	.003	.001

Table VI: Kruskal-Wallis test.

Time distribution of the identified patterns of behavior is illustrated in Figure 2. The horizontal axe represents months from January 2001 to December 2006. 60 out of these 72 observations have been categorized in one of the three latent factors obtained. Although the months associated to each factor are approximately 1/3 of the total, their distribution over time is not the same. The months associated with factor 1, a poor debate, tend to be grouped, while the months associated to factors 2 and 3 tend to alternate among them.

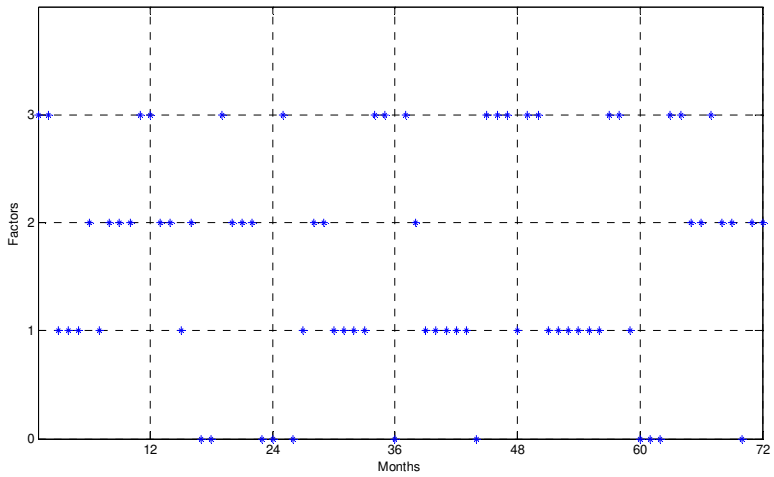

Figure 2: Time evolution of ARM Embedded Linux mailing list.

The obtained results conclude that the mailing list contributes in general to the successful evolution of Linux-based ARM processors, but the behavior of the list is not always the same. Approximately, two-third of the time the list exhibits good health represented by factors 2 y 3. But on-third of the time, the list is not answering developers' expectations. Besides, the behavior described by factor 1 tends to be extended in time, as it is shown in Figure 2. The result is very problematic in this case, because those developers who are not obtaining an answer may decide to change and look for new sources of support. This troublesome situation is usually inverted with a change to factor 3, as it can be observed during the years 2003, 2004 and 2005 in Figure 2. That means that the core of the community is coming to the list aid. The

presence of an involved core of individuals with a strong commitment to the community is essential to achieve a good behavior and guarantying that the list is fulfilling its aims.

6 Conclusions

The purpose of this paper has been the characterization of mailing list behavior in OSS projects. Three kinds of behaviors in the case study of ARM Linux mailing list have been identified. The first kind of behavior is quite negative, as a lot of posts are not answered, causing a frustration effect among developers because the mailing list is not working properly. The second identified behavior represents a real community of practice, with active authors and a variety of threads. Mailing list can be considered in this case a very good source of support for developers. The last obtained behavior is a hyper concentrated debate around a few questions. Although such kind of debate is not negative, a prolonged behavior like that may cause the absence of a variety of threads, and subsequently authors will finally feel themselves also frustrated.

The sample has been organized per months to allow a longitudinal analysis. Basically, the presence of the mentioned behaviours through the last six years has been detected. The evolution illustrates that list not always exhibit a positive behavior and it is necessary a monitoring activity from the core of the community to solve this kind of situations.

Acknowledgements

This work has been supported by the Spanish Ministry of Education and Science (Research Project with reference DPI2007-60128) and the Consejería de Innovación, Ciencia y Empresa (Research Project with reference P07-TIC-02621)

References

[Abbott, 2003] Abbott, D.: Linux for Embedded and Real Time Applications. Newnes, Elsevier Science, USA, 2003.

[Aguila-Obra and Padilla-Meléndez, 2006] Aguila-Obra, A., Padilla-Meléndez, A.: Organizational factors affecting Internet technology adoption, Internet Research, Vol. 16, no. 1, pp. 94-110, 2006.

[Applewhite, 2003] Applewhite, A.: Should governments go Open Source?, IEEE Software, Vol. 20, no. 4, pp. 88–91, 2003.

[Barrero et al., 2008] Barrero, F., Toral, S.L., Gallardo, S.: eDSPLab: Remote Laboratory for Experiments on DSP Applications, Internet Research, Vol. 18, no. 1, pp. 79-92, 2008.

[Bird et al., 2006] Bird, C., Gourley, A., Devanbu, P., Gertz, M., Swaminathan, A.: Mining Email Social Networks, Proceedings of the International Workshop on Mining Software Repositories, MSR'06, May 22–23, Shanghai, China, pp. 137-143, 2006.

[Boland and Tenkasi, 1995] Boland, R., Tenkasi, R.: Perspective making and perspective taking in communities of knowing, Organization Science, Vol. 6, no. 4, pp. 350–372, 1995.

[Bonaccorsi and Rossi, 2003] Bonaccorsi, A., and Rossi, C.: Why Open Source Software can succeed, Research Policy, 32, pp. 1243–1258, 2003.

[Bonacci, 2004] Bonacci, D.: Towards quantitative tools for analysing qualitative properties of virtual communities. Interdisciplinary Description of Complex Systems, Vol. 2, no. 2, pp. 126-135, 2004.

[Brown and Duguid, 1998] Brown, J.S., Duguid, P.: Organizing knowledge, California Management Review Spring, Vol. 40, no. 3, pp. 90–106, 1998.

[Dahlander and Magnusson, 2005] Dahlander, L., Magnusson, M.G.: Relationships between open source software companies and communities: Observations from Nordic firms, Research Policy, Vol. 34, no. 4, pp. 481–493, 2005.

[Dinh-Trong and Bieman, 2005] Dinh-Trong, T.T., and Bieman, J.M.: The FreeBSD Project: A Replication Case Study of Open Source Development, IEEE Transactions on Software Engineering, Vol. 31, No. 6, pp. 481-494, 2005.

[Droschl, 2004] Droschl, G.: Communities of Practice: An Integrated Technology Perspective, Journal of Universal Computer Science, vol. 10, no. 3, pp. 284-293, 2004

[Dyba, 2005] Dyba, T.: An Empirical Investigation of the Key Factors for Success in Software Process Improvement, IEEE Transactions on Software Engineering, Vol. 31, No. 5, pp. 410-424, 2005.

[Free Software Foundation, 2001] Free Software Foundation: GNU General Public License, 2001, available at: www.fsf.org/licenses/gpl.html.

[Gambardella and May, 2006] Gambardella, A., May, B.H.: Proprietary versus public domain licensing of software and research products, Research Policy, 35, pp. 875–892, 2006.

[Ganssle and Barr, 2003] Ganssle, J., and Barr, M.: Embedded Systems Dictionary, Lawrence, KS: CMP Books, 2003.

[Garrety et al., 2004] Garrety, K., Robertson, P.L., Badham, R.: Integrating communities of practice in technology development projects, International Journal of Project Management, 22, pp. 351–358, 2004.

[Gorsuch, 1974] Gorsuch, R.L.: Factor Analysis. Philadelphia: W.B. Saunders Co, 1974.

[Gruber & Henkel, 2006] Gruber, M., Henkel, J.: 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, 2006.

[Gutwin et al., 2004] Gutwin, C., Penner, R., Schneider, K.: Group awareness in distributed software development, Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work, Chicago, pp. 72–81, 2004.

[Henkel, 2003] Henkel, J.: Software development in embedded Linux - informal collaboration of competing firms, Proceedings der 6. Internationalen Tagung Wirtschaftsinformatik, Dresden, pp. 1-20, 2003.

[Henkel, 2006] Henkel, J.: Selective revealing in open innovation processes: The case of embedded Linux, Research Policy 35, pp. 953–969, 2006.

[Hertel et al., 2003] Hertel, G., Niedner, S., and Herrmann, S.: Motivation of software developers in Open Source projects: An Internet-based survey of contributors to the Linux kernel, Research Policy, 32, pp. 1159–1177, 2003.

[Johnson, 2006] Johnson, J.P.: Collaboration, peer review and open source software, Information Economics and Policy, 18, pp. 477–497, 2006.

[Jones, 2004] Jones, G., Ravid, G., Rafaeli. S.: Information overload and the message dynamics of online interaction spaces: A theoretical model and empirical exploration, Information Systems Research, Vol. 15, pp. 194–210, 2004.

[Kogut & Metiu, 2001] Kogut, B., and Metiu, A.: Open Source Software and distributed innovation, Oxford Review of Economic Policy, Vol. 17, no. 2, pp. 248–264, 2001.

[Kolbitsch & Maurer, 2006] Kolbitsch, J. Maurer, H.: The Transformation of the Web: How Emerging Communities Shape the Information we Consume, Journal of Universal Computer Science, vol. 12, no. 2, pp. 187-213, 2006

[Krogh et al., 2003] Krogh, G., Spaeth, S., Lakhani, K.: Community, joining, and specialisation in open source software innovation: a case study, Research Policy, 32, pp. 1217–1241, 2003.

[Kuk, 2000] Kuk, G.: When to speak again: Self-regulation under facilitation, Group Dynamics: Theory, Res. Practice, 4, pp. 291–306, 2000.

[Lakhani & Hippel, 2003] Lakhani, K., Hippel, E.: How open source software works: 'free' user to user assistance, Research Policy, 32, pp. 923–943, 2003.

[Lakhani & Wolf, 2005] Lakhani, K.R., Wolf, R. G.: Why hackers do what they do: Understanding motivation effort in free/open source software projects, In: Feller, Joseph, Fitzgerald, Brian, Hissam, Scott A., Lakhani, Karim R. (Eds.), Perspectives on Free and Open Source Software. MIT Press, Cambridge, pp. 3–21, 2005.

[Li et al., 2006] Li, X., Montazemi, A.R., Yuan, Y.: Agent-based buddy-finding methodology for knowledge sharing, Information & Management, 43, pp. 283–296, 2006.

[Little et al., 2003] Little, R. G. Jr., and Gibson, M.L.: Perceived Influences on Implementing Data Warehousing, IEEE Transactions on Software Engineering, Vol. 29, No. 4, pp. 290-296, 2003.

[Madanmohan and Navelkar, 2004] Madanmohan, T.R., and Navelkar, S.: Roles and knowledge management in online technology communities: an ethnography study, International Journal of Web Based Communities, Vol. 1, No. 1, pp. 71-89, 2004.

[Madey et al., 2004] Madey, G., Freeh, V., Tynan, R.: Modeling the F/OSS community: A quantitative investigation, S. Koch, ed. Free/Open Source Software Development. Idea Publishing, Hersey, PA, pp. 203–220, 2004.

[Martínez-Torres and Toral, in press] Martínez-Torres, M.R., Toral, S.L.: International Comparison of R&D Investment by European, US and Japanese Companies, International Journal of Technology Management, in press.

[McDonald et al., 2003] McDonald, C.J., Schadow, G., Barnes, M., Dexter, P., Overhage, J.M., Mamlin, B.: Open Source Software in medical informatics—why, how and what, International Journal of Medical Informatics, Vol. 69(2/3), pp. 175–184, 2003.

[Mockus et al., 2002] Mockus, A., Fielding, R.T., Herbsleb, J.D.: 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, 2002.

[Niedner et al., 2000] Niedner, S., Hertel, G., Hermann, S.: Motivation in Open Source Projects: an Empirical Study among Linux Developer, Summarized study results available at: http:/www.psychologie.uni-kiel.de/linux-study, 2000.

[Nowak & Wurst, 2004] Nowak J., Wurst M.: Supporting Knowledge Creation and Sharing in Communities Based on Mapping Implicit Knowledge, Journal of Universal Computer Science, vol. 10, no. 3, pp. 235-251, 2004

[Perens, 1997] Perens, B.: The Open Source definition, available at: http://www.opensource.org/docs/definition.php, 1997.

[Perkins, 1999] Perkins, G.: Cultural Clash and the Road to World Domination, IEEE Software, vol. 16, no. 1, pp. 23-25, 1999.

[Preece, 2001] Preece, J.: Sociability and usability: twenty years of chatting online, Behaviour and Information Technology Journal, Vol. 20, no. 5, pp. 347–356, 2001.

[Raghavan et al., 2006] Raghavan, P., Lad, A., Neelakandan, S.: Embedded Linux System Design and Development, Auerbach Publications, Taylor and Francis Group, 2006.

[Raymond, 1999] Raymond, E.S.: The cathedral and the bazaar, available at http://www.tuxedo.org/~esr/writings/cathedral-bazaar/, 1999.

[Rencher, 2002] Rencher, A.C.: Methods of Multivariate Analysis, 2nd ed. Wiley Series in Probability and Statistics. John Wiley & Sons, 2002.

[Roberts et al., 2006] Roberts, J.A., Hann, I.L., Slaughter, S.A.: Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects, Management Science, Vol. 52, No. 7, pp. 984–999, 2006.

[Sowe et al., 2006] Sowe, S., Stamelos, I., Angelis, L.: Identifying knowledge brokers that yield software engineering knowledge in OSS projects, Information and Software Technology, 48, pp. 1025–1033, 2006.

[Spinellis & Szyperski, 2004] Spinellis, D., and Szyperski, C.: How is Open Source affecting software development?, IEEE Software, Vol. 21, no. 1, pp. 28–33, 2004.

[Spinellis, 2006] Spinellis, D.: Open Source and Professional Advancement, IEEE Software, 23, 5, pp. 70–71, 2006.

[Valimaki & Oksanen, 2005] Valimaki, M., Oksanen, V.: The impact of free and open source licensing on operating system software markets, Telematics and Informatics, 22, pp. 97–11, 2005.

[Waring & Maddocks, 2005] Waring, T., Maddocks, P.: Open Source Software implementation in the UK public sector: Evidence from the field and implications for the future, International Journal of Information Management, 25, pp. 411-428, 2005.

[Wenger, 1998] Wenger, E.: Communities of practice: learning, meaning, and identity, Cambridge: Cambridge University Press, 1998.

[Xu et al., 2005] Xu, J., Gao, Y., Christley, S. and Madey, G.: A Topological Analysis of the Open Source Software Development Community. Proceedings of the 38th Annual Hawaii International Conference on System Sciences. HICSS '05.,188-198, 2005.

[Yaghmour, 2003] Yaghmour, K.: Building Embedded Linux Systems, O'Reilly, 2003.

[Zeitlyn, 2003] Zeitlyn, D.: Gift economies in the development of open source software: Anthropological reflections, Research Policy 32, pp. 1287–1291, 2003.