A genetic search of patterns of behaviour in OSS communities

M.R. Martínez-Torres *

University of Seville, Facultad de Turismo y Finanzas, Avda. San Francisco Javier, s/n 41018 Sevilla, Spain

ARTICLE INFO

Keywords:
Open source software
Virtual communities
Social network analysis
Genetic algorithm
Factor analysis

ABSTRACT

This paper proposes the identification of patterns of behaviour of open source software (OSS) communities using factor analysis and their social network analysis (SNA) features. OSS communities can be modelled as a social network in which nodes represent the community members and arcs represent the social interactions among them, and factor analysis is able to provide the factors that explain the latent patterns of behaviour. Due to the complexity of the problem and the high number of SNA features that can be extracted, this paper proposes a genetic search of an optimum subset of indicators leading to a group of latent patterns of behaviour maximizing the explained data variance and the interpretation of factors. Obtained results illustrate the feasibility of the proposed framework to extract relevant information from a large set of data.

1. Introduction

The OSS projects differ greatly from commercial software development models in several aspects. For instance, commercial software companies prevent access to the source code of their products from outside developers and customers, while OSS allows source code to be freely modified and redistributed under "open source" licenses (Feller & Fitzgerald, 2002). Besides, OSS projects are typically developed in a distributed and decentralized way as a difference to proprietary software, based on closed and formal structures. Precisely, one of the most distinctive characteristics of OSS projects is the fact that they are written, developed, and debugged largely by worldwide volunteers, who in most cases are connected and collaborate solely through the Internet. Therefore, the community behind the development of the project plays an essential role for the project to success (Deek & McHugh, 2008).

Several authors have described OSS development teams as having a hierarchical or onion-like structure (Crowston & Howison, 2005), with a central core of highly active individuals, surrounded by other layers of progressively less active individuals. One example of this is presented in the study by Ye, Nakakoji, et al. (2005) where the central core is composed of the project leaders and core members, with five outer layers containing active developers, peripheral developers, bug reporters, passive users, and stakeholders, respectively. It has been demonstrated that much of the OSS development is realized by a small percentage of individuals despite the fact that there are tens of thousands of available developers. Such concentration is called "participation inequality" (Kuk, 2006), and it can be explained by the different user profiles of open source communities. Participation inequality allows the categorization of OSS community members in three groups (Mockus, Fielding, & Herbsleb, 2002; Xu, Gao, Christley, & Madey, 2005). Core members 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. Moderators and leaders are included in this group. Active developers are those community members that regularly make contributions to the project. Finally, peripheral developers 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 social structure of OSS teams directly influences the participation and the decision-making process affecting the overall performance of the project. Therefore, an important research question is extracting the different patterns of behaviour in OSS communities. These patterns leading to successful projects are of great interest both for autonomous and sponsored communities, that share a common aim of retaining and attracting participants to their communities (West & O'mahony, 2008). However, the structure of communities can only be derived from the participation activity of their members. For this purpose, OSS communities have been frequently modelled as a social network, being the nodes of the network the community members while the arcs represent the flow of interactions among users (Toral, Martínez-Torres, & Barrero, 2009a). These networks are then analyzed using Social Network Analysis (SNA) techniques by obtaining a set of SNA features. For instance, previous studies have considered the size and out-degree of nodes (Valverde, Theraulaz, Gautrais, Fourcassie, &

* Tel.: +34 954 55 43 10; fax: +34 954 55 69 89.
E-mail address: rmtorres@us.es

Sole, 2006), closeness centrality (Panchal, 2009) and betweeness centrality (Hossain, Wu, & Chung, 2006; Toral, Martínez-Torres, Barrero, & Cortés, 2009b), the clustering coefficient (Kwon, Oh, & Jeon, 2007; Singh, 2010), structural holes (Okoli & Oh, 2007) or the brokerage role of nodes (Sowe, Stamelos, & Angelis, 2006; Barcellini, Détienne, & Burkhardt, 2009; Toral, Martínez Torres, & Barrero, 2010), among other SNA features. Moreover, each of these measurements can be computed for the whole network or for several specific sub-networks like the one obtained from active developers or from the core group of the community. Due to the large number of possible SNA indicators that can be obtained, previous studies have been focused on a small number of social network features to characterize participation and obtain OSS patterns of behaviour. As a difference, in this study participation is analyzed using the whole set of SNA features that can be obtained from OSS communities. For this purpose, a genetic search of the optimum subset of indicators able of providing the main pattern of behaviour in OSS communities is proposed. Therefore, the main contribution of this paper is the possibility of dealing with all the SNA features of the social networks modelling OSS communities through an evolutionary computation technique like Genetic Algorithms.

The remainder of the paper is structured as follows. First, previous studies related to social network structures in OSS communities are reviewed in Section 2. In Section 3 the problem is formulated and the proposed approach described. Section 4 describes the Genetic Algorithm implementation. Obtained results and discussion are included in Section 5. Finally, conclusions are detailed in Section 6.

2. Related work

Social network theory is based on the idea that social interaction patterns reflect the behaviour of individuals (Freeman, 2004). Therefore, the analysis of the social interactions can provide information about how individuals behave or how groups are organized. This is why social network analysis focuses on the relationships between people, instead of on characteristics of people. By mapping these relationships, network analysis helps to uncover the emergent and informal communication patterns present in an organization, which in turn can be used to explain several organizational phenomena.

A social network can be modelled as a graph with nodes representing people or groups, and links representing relationships or information flows between them. OSS communities constitute clear examples of dynamic social networks, as it is changing over time. Social networks begin when developers join a project, work with others, and form co-working relationships (Xu, Christley, & Madey, 2006). These relationships are important because the sense of belonging to a group is encouraged through social network bonding. The aim of OSS communities is attracting and retaining people, as this will benefit the underlying software. As new members join the community, different user profiles emerge. Not all the users are interested in participating the same way. Some of them, a small percentage, intensely contribute to the development of the project, while the rest of them make regular, occasional or even no contribution at all. These variety of user profiles lead to a core/periphery structure, where the core group of developers is located at the center of the community and the rest of them far away from the center depending on their contributions. The core group members are strongly connected to each other while the periphery contains members who are usually weakly connected to each other as well as to the core members (Long & Siau, 2007).

Mathematically, a social network can be represented as a graph $G = (V, \mathrm{E})$ where V denotes a finite set of vertices and E denotes a finite set of edges such that $E \subseteq V \times V$. Some network analysis methods are easier to understand when graphs are conceptualized as matrices (Nooy, Mrvar, & Batagelj, 2005) as shown in Eq. (1).

$$M = (m_{ij})_{n^*n}, \quad \text{where } n = |V|, \ m_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

In case of a valued graph, real valued weight function $w(e)$ is defined on the set of edges, i.e., $w(e) = Ex\Re$, and the matrix is then defined as given by Eq. (2).

$$M_{i,j} = \begin{cases} w(e) & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

In the context of OSS communities, V is given by all the community members and E is given by the interactions among them. The resulting network is a directed network in the sense that edges are actually arcs from one community member to another one, and the direction of the arcs shows the flow of information between them. It is also a valued network, as it is possible multiple interaction between the same community members.

Networks can be partitioned using some discrete characteristics of vertices. For instance, several classes of vertices can be obtained using the function $w(e)$, that is, the strength of arcs. In the case of OSS projects, these kinds of partitions should highlight the core/periphery (C/P) structure of the community. A C/P structure divides vertices in three distinct subgroups: vertices in the core, densely connected with each other, and vertices on the periphery, which in turn can be divided in active and peripheral vertices depending on their level of interaction.

The following list summarizes the characteristics that can be computed for the whole network or each of the mentioned subnetworks as well as the main previous studies focused on them.

Size and connectivity: the size of the community is the number of members the community has and the number of arcs represent the connectivity among these community members. Both indicators has been frequently used as a measure of the successful development of a OSS project. Success in a virtual community could be manifested through the level of participation, which can be understood as the number of participants (Preece, 2001) or as the level of community activity and quantity of community work output (Hinds & Lee, 2008). Several topological indices based on connectivity can be computed, like the Zagreb group index, the Randic connectivity index or the Platt index, all of them defined in terms of connections of nodes (Devillers & Balaban, 1999).

Density: it is defined as the number of lines in a simple network, expressed as a proportion of the maximum possible number of lines. The main problem of this definition is that it does not take into account valued lines higher than 1 and it depends on the network size. A different measure of density is based on the idea of the degree of a node, which is the number of lines incident with it (Toral et al., 2009a). A higher degree of nodes yields a denser network, because nodes entertain more ties. The advantage of average degree is that it is a non-size dependent measure of density. As OSS communities are directed networks, several statistical measures of the out-degree distribution can be considered. Finally, density can be measured alternatively using an egocentric point of view; the egocentric density of a node is the density of ties among its neighbours (Nooy et al., 2005). In previous works, density has been used to study the coordination performance of OSS projects. Several studies conclude there is a negative impact of density over the quality of the project and its coordination (Hossain & Zhu, 2009; Feczak & Hossain, 2011).

Components: A strong component is a maximal strongly connected subnetwork. A network is said to be strongly connected if each pair of vertices is connected by a path, taking into account the direction of arcs (Nooy et al., 2005). In the context of this study,

components allow the identification of connected substructures in the OSS community.

K-cores: a *k*-core is a sub-network in which each node has *k* degree in that sub-network. The core with the highest degree is the central core of the network, detecting the set of nodes where the network rests on (Toral et al., 2010).

Distance: it is defined as the number of steps in the shortest path that connects two vertices. Distance between members in the OSS community can affect how ideas and discussions can spread over the community. Short distances mean information only has to travel a few links to reach anybody in the network (Xu et al., 2006).

Closeness centralization: it is an index of centrality based on the concept of distance. The closeness centrality of a node is calculated considering the total distance between one node and all other nodes, where larger distances yield lower closeness centrality scores. The closeness centralization is an index defined for the whole network, and it is calculated as the variation in the closeness centrality of vertices divided by the maximum variation in closeness centrality scores possible in a network of the same size (Toral et al., 2009b). It has been found that closeness centralization has a positive correlation with coordination mechanism on OSS projects (Pereira & Soares, 2007; Feczak & Hossain, 2011).

Betweenness centrality: it is a measure of centrality that rests on the idea that a person is more central if he or she is more important as an intermediary in the communication network (Nooy et al., 2005). The centrality of a node depends on the extent to which this node is needed as a link to facilitate the connection of nodes within the network. If a geodesic is defined as the shortest path between two nodes, the betweenness centrality of a vertex is the proportion of all geodesics between pairs of other vertices that include this vertex, and betweenness centralization of the network is the variation in the betweenness centrality of vertices divided by the maximum variation in betweenness centrality scores possible in a network of the same size. It has been used to study the hierarchy and centralization in free and open source software team communications (Crowston & Howison, 2006).

Brokerage roles: A broker is a middle node in a directed triad (a set of three vertices and the lines among them). Different types of brokerage roles can be distinguished considering mediation between members of the same or different groups. In this context of OSS communities, these groups are given by active and peripheral members. Therefore, two possibilities of mediation can be considered as shown in Fig. 1.

The role of knowledge brokers has been highlighted in mailing lists as community facilitators, helping answer those questions knowledge seekers posted (Sowe et al., 2006). This role has also been assigned to the core team of the OSS project, acting as intermediary between expert software developers and peripheral users and helping OSS projects to engage in a discourse and co-learning experience with their user communities (Toral et al., 2010).

Clustering coefficient: It measures whether first degree neighbours of a particular vertex interact with each other. Basically, clustering coefficient is a measure of local cohesiveness through the neighbour interactions of a vertex (Durugbo, 2012). The clustering coefficient of a vertex is defined as the ratio of the number of links to the total possible number of links among its neighbours, and the clustering coefficient of a social network is the average of all the clustering coefficients of the vertices. It has been used to characterize the small-world phenomena in networks (Xu et al., 2006). Besides, highly clustered networks provide better information propagation (Gao & Madey, 2007).

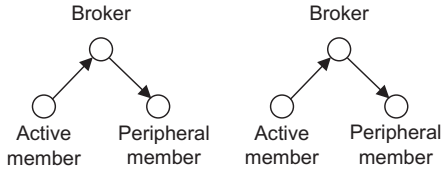

Fig. 1. Brokerage roles.

Proximity prestige: It is defined in terms of the output domain of a node, which is the number of nodes for which there is a path to that node. Following Nooy et al. (2005) definition, the proximity prestige of a node is the proportion of all nodes in the network (excluding itself) that are in its output domain divided by the mean distance from all nodes in its output domain. Therefore, a zero value of the proximity prestige means that this node is isolated.

3. Formulation of the problem and proposed framework

The problem consists of finding a set of SNA indicators able to explain the different structural patterns of OSS communities. Factor Analysis is a multivariate statistical technique usually employed for the identification of latent dimensions or factors on a dataset. These factors are not directly observable and segment the dataset into relatively homogeneous segments (Rencher, 2002). It is assumed each variable is dependent on a linear combination of the common factors, and the coefficients are known as loadings (Toral & Martínez Torres, 2009c). Mathematically, the factor analysis model expresses each variable as a linear combination of underlying common factors $f_1, f_2, \ldots, f_m$, with an accompanying error term to account for that part of the variable that is unique (not in common with the other variables). For $y_1, y_2, \ldots, y_p$ in any observation vector y, the model is as follows:

$$\begin{aligned} y_1 - \mu_1 &= \lambda_{11} f_1 + \lambda_{12} f_2 + \cdots + \lambda_{1m} f_m + \varepsilon_1 \\ y_2 - \mu_2 &= \lambda_{21} f_1 + \lambda_{22} f_2 + \cdots + \lambda_{2m} f_m + \varepsilon_2 \\ &\ldots \\ y_p - \mu_p &= \lambda_{p1} f_1 + \lambda_{p2} f_2 + \cdots + \lambda_{pm} f_m + \varepsilon_p \end{aligned} \tag{3}$$

Model (3) can be written in matrix notation as in Eq. (4), where Λ is the factor loadings matrix.

$$y - \mu = \Lambda f + \varepsilon \tag{4}$$

The coefficients λ_{ij} are called loadings and serve as weights, showing how each y_i individually depends on the underlying factors (Lee and Lee, 2011). With appropriate assumptions, λ_{ij} indicates the importance of the *j*th factor f_j to the *i*th variable y_i and can be used in interpretation of f_j. It is expected the loadings will partition the variables into groups corresponding to factors.

Ideally, the number of factors *m* should be substantially smaller than the problem dimension *p*; otherwise we have not achieved a parsimonious description of the variables as functions of a few underlying factors. In the case of exploratory factor analysis, the lack of theoretical background causes that the number of factors to be extracted is a priori unknown. Therefore, factors must be selected attending to the homogeneity of their indicators.

The main problem of using factor analysis when considering a large set of indicators is that the final result is conditioned by the number of variables included in the analysis. The described model try to fit the data in set of factors, and the inclusion of non appropriate variables may distort the obtained latent factors. However, it is not easy to decide which variables are or not appropriate, above all in those situations where the theoretical background cannot guide this process. This paper proposes a computation framework where the selection of indicators is performed using Genetic Algorithms (GA). Each element (or chromosome using typical GA notation) of the population considered by GA represent a subset of all the possible SNA indicators. Fig. 2 shows a brief scheme of the proposed framework.

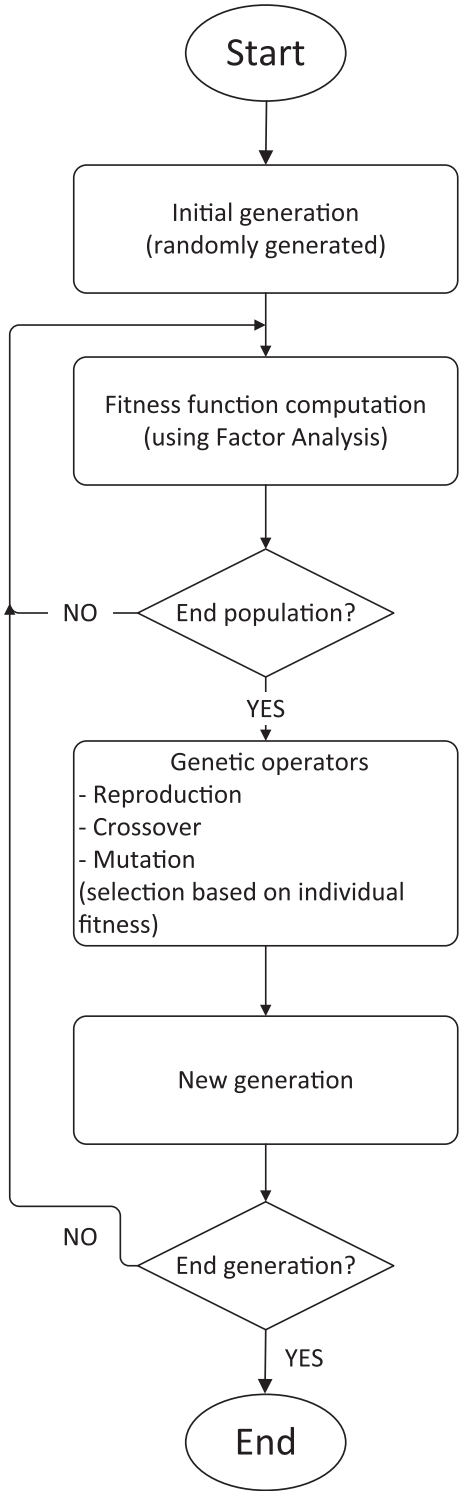

Fig. 2. Proposed framework.

The developed algorithm consists of two loops. The inner loop consists of the evaluation of the fitness function for each element of the population using factor analysis. The outer loop generates a new generation using genetic operators like reproduction, crossover and mutation, using a selection based on the individual fitness of each element of the population. The framework stops working when a selected stopping criterion is reached.

4. Genetic algorithm implementation

Genetic Algorithms are a family of computational models inspired by evolution (Holland, 1975; Goldberg, 1989). These algorithms encode a potential solution to specific problem on a simple chromosome-like data structure and apply genetic operators to these structures in order to preserve critical information (Martínez-Torres & Toral-Marín, 2010). An initial population Pi composed of N_t chromosomes is considered. Goldberg (1989) studied the optimum number of chromosomes for a population according to the chromosome's length. His main conclusion was that the optimum population's size value gets higher as the chromosome's length increases. This initial population is generated randomly in order to preserve the diversity in the population and the fitness function is calculated to evaluate the goodness of each chromosome. The mechanism for generating the subsequent generations is based on the selection scheme from $(\mu + \lambda)$ evolution strategy (Michalewicz, 1996; Reina, Toral, Johnson, & Barrero, 2012). The μ best chromosomes are included directly in the next generation. The crossover and mutation operations are responsible of generating λ chromosomes of a new population. The crossover consists of using two members of a population P_j to generate two new members of the next population P_{j+1} by crossing their genetic information. The new chromosomes contain genetic information from the predecessors. The purpose of mutation is to change the genetic information of a chromosome included in P_j to generate a new chromosome of P_{j+1}. Fig. 3 illustrates the implemented genetic algorithm.

4.1. Chromosome encoding

A chromosome C_i represent the subset of indicators that will be considered to perform factor analysis. The total number of

```
BEGIN /* Genetic Algorithm */
j=0
Generate Pj
WHILE (Not stop criteria) DO
Evaluate Pj
BEGIN /* Generate next generation */
        Select the μ best of Pj
        Randomly selection for crossing
        Crossover (Pc)
        Mutation (Pm)
        Create Pj+1
        j=j +1
        END
END
```

Fig. 3. Outline of the genetic algorithm implementation.

indicators extracted for the OSS communities according to the described SNA features is 60 (see appendix). Therefore, each chromosome is a binary string of length 60 where a value of 1 means that the corresponding variable is part of the considered subset of indicators and a value of 0 means that it is excluded from this list. Notice that the space of possible solutions is formed by 2^{60} = 1.1529e+018 possibilities. That means that we should perform 2^{60} different factor analyses to completely explore the space of possible solutions. In this kind of problems, GA can perform a guided search of the optimum solution with lower computational cost than exploring one by one all the possibilities. The representation of chromosomes as binary strings has the advantage that this chromosomal encoding is complete and valid. Complete means that the whole space of possible solutions can be represented and valid means that all of them can be computed. The composition of a chromosome is shown in Fig. 4.

4.2. Evaluation function

The fitness function quantifies the suitability of each chromosome as a solution. Genetic operators make selections based on individual fitness. That means that chromosomes with high fitness value have more chance of being selected, passing their genetic material (via reproduction, crossover or mutation) to the next generation. As a result, the fitness function provides the pressure for evolution towards a new generation with chromosomes of higher fitness than the previous ones. In this case, the fitness function should measure how well factor analysis can identify latent factors. However, its capacity to perform such identification depends on the following parameters:

- Explained variance: It refers to the percentage of the total sample variance explained by the considered factors.
- Correlations between variables: It is the average of the sum of the squared correlation coefficients between indicators. Considered indicators must be correlated as the factor analysis is based on the interrelationships among variables.
- Interpretability of factors: A factor is well defined if it is explained by at least three variables (Rencher, 2002). That is to say, at least three different factor loadings should be maximized for each considered factor.

As a result, the fitness function requires to be defined as a multi-objective fitness function considering the aforementioned parameters.

$$F = c_1\ \mathrm{Var} + c_2 \frac{1}{n}\sum_{i=1}^{k} r_i^2 + c_3\ \mathrm{Interp} \tag{5}$$

Explained variance and correlations among variables exert opposite effects on the evolution of GA. Explained variance makes the GA to evolve towards a minimum number of indicators, as it is easiest to explain the variance of the dataset whenever a lower number of variables are considered. As a difference, correlations among variables makes the GA to search for a higher number of variables, as this parameter is maximized by including as many variables as possible. However, the third parameter, interpretability of factors, is the most important one, because this parameter guarantees factors are well defined.

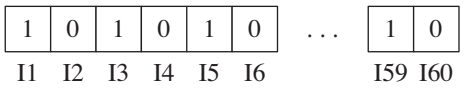

Fig. 4. Chromosome's composition.

C_1, C_2, and C_3 coefficients in Eq. (5) are used to adjust the relative importance of the three parts of the fitness function. Obviously, the range of them is [0,1], with the restriction of $C_1 + C_2 + C_3 = 1$.

4.3. Stopping criteria

The population's average fitness function has been chosen as the stop criterion of the genetic algorithm. If $P_{\mathrm{av},j}$ denotes the population's average fitness function, the stopping criterion can be formulated as:

$$S_c \rightarrow P_{\mathrm{av},j+1} \leq P_{\mathrm{av},j} \tag{6}$$

4.4. Procedure of transition

The procedure used to generate a new population P_{j+1} from the previous population P_j is as follows:

- The best 20% chromosomes are copied from P_j to P_{j+1}. This ensures that the best individuals of each population will be included in the next generation. As a consequence, the likelihood of using a good chromosome for reproduction operations gets higher.
- The 80% of the new chromosomes are generated by using crossover (75%) and mutation (5%) operations. This aims to favour the diversity of the chromosomes. P_c denotes the probability of a chromosome C_i to take part in a crossover operation and it is calculated as:

$$P_{Ci} = \frac{f_f(C_i)}{\sum_{i=1}^{n} f_f(C_i)} \tag{7}$$

The term $f_f(C_i)$ stands for the evaluation of the fitness function for the chromosome C_i. Consequently, the best chromosomes are more likely to be selected. The crossover operation is illustrated in Fig. 5.

A one-point crossover operation has been implemented. The point of cross is denoted by p_k, where $0 \leqslant k \leqslant l$, and l is the size of the chromosome. The value of k is randomly chosen for each crossover operation. This point divides each chromosome into two parts RG_j and LG_j. The two new chromosomes are then obtained swapping $LG_{j,1}$ by $LG_{j,2}$.

Similarly, P_m is the probability of a chromosome i to take part in a mutation operation. The purpose of mutation is to make small changes in the chromosomes. These changes consist of modifying one chromosome's bit. According to De Jong (1975) we have calculated P_m as l^{-1}. The mutation operation is illustrated in Fig. 6.

The position of the mutated bit is denoted by p_m, where $0 \leqslant m \leqslant l$. The value of p_m is randomly chosen for each mutation operation.

5. Results

The proposed approach has been applied to twelve virtual communities listed in Table 1. They correspond to Linux Debian ports to different processor architectures. The Debian Project is an association of individuals who have made common cause to create a free operating system called Debian GNU/Linux, or simply Debian for short (Robles, Gonzalez-Barahona, & Michlmayr, 2005; Mateos-Garcia & Steinmueller, 2008).

Each community was analyzed from the year in which each community started its activity to 2010. For each year and community, a social network based on interactions among participants has been extracted. As a result, a total of 134 social networks have been analyzed, extracting the set of data detailed in the appendix.

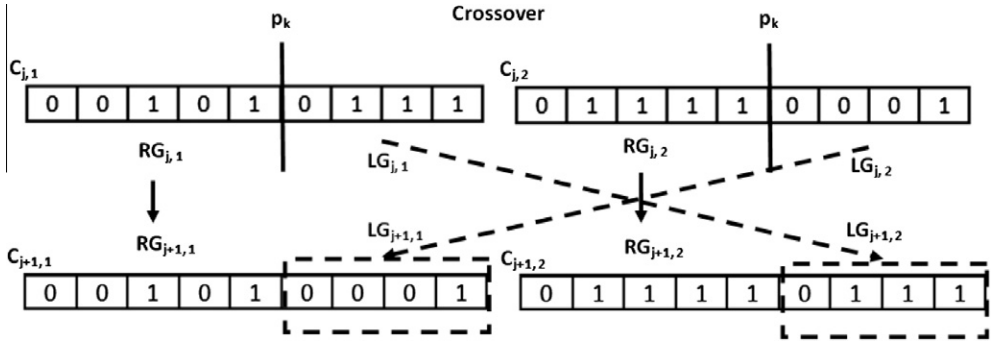

Fig. 5. Crossover operation.

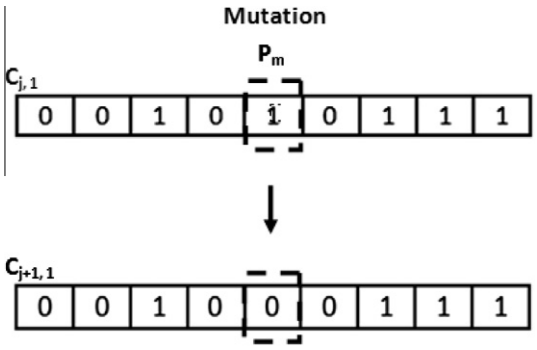

Fig. 6. Mutation operation.

Once this large dataset has been obtained, GA has been applied to obtain an optimum subset of indicators able to identify OSS communities profiles according to their topological participation structure. A population's size of 10000 chromosomes will be considered (Alander, 1992). That means that the 2000 best chromosomes are directly included in the next population, 7500 chromosomes are generated by the crossover operation and 500 chromosomes are generated by the mutation operation. Factor analysis is used for evaluating each chromosome. The cost function follows the general structure defined in Eq. (5). The coefficients C_1, C_2 and C_3 represent the relative weigh of each part of the fitness function. GA has been run for different values of the three parameters, obtaining the results shown in Table 2. It can be observed that interpretability only reaches a value of 1 when C_3 is overweighed, while the explained variance varies depending on the value of C_1. That is why the set of coefficients with the values $C_1 = 0.1$, $C_2 = 0.1$ and $C_3 = 0.8$ has been chosen (This set of value is shown in italics in Table 2). An interpretability of factors equal to 1 guarantees that all the latent factors are interpretable. Using this set of values, GA converged after 30 generations, with an explained variance of 70.93%, and 23 indicators grouped in four factors. Time required by genetic algorithm execution was 2873.6 s (47.89 min). This value is much smaller than the alternative option of exploring the whole solution space. According to the chosen encoding, the size of a chromosome is $l = 60$ *bits*, so the space of possible solutions is $2^l = 2^{60} = 1.1529e{+}018$. The idea of finding the optimum solution exploring all the possibilities is unattainable. The time necessary to carry out a single factor analysis is 12.9 ms. That means it would take more than 470 million years to explore the whole space of possible solutions. The genetic algorithm implementation is able to speed up the search of an optimal solution.

Table 1
Analyzed Linux Debian communities.

	URL	Description	Period
Debian port to m68k (Debian-68k)	http://lists.debian.org/debian-68k/	Motorola 68k port of Debian GNU/Linux. Debian currently runs on the 68020, 68030, 68040 and 68060 processors	1998–2010
Debian port to ARM (Debian-ARM)	http://lists.debian.org/debian-arm/	ARM port for Debian GNU/Linux. Debian fully supports a port to little-endian ARM	1999–2010
Debian port to Intel IA-64 (Debian-ia64)	http://lists.debian.org/debian-ia64/	Discussions on the intel IA64 (aka Itanium, Merced) port of Debian GNU/Linux	2001–2010
Debian port to Alpha (Debian-alpha)	http://lists.debian.org/debian-alpha/	The purpose of this project is to assist developers and others interested with the ongoing project to port the Debian distribution of Linux to the Alpha family of processors	1998–2010
Debian port to AMD64 (Debian-amd)	http://lists.debian.org/debian-amd64/	Porting Debian to AMD x86-64 architecture	2004–2010
Debian port to BSD (Debian-bsd)	http://lists.debian.org/debian-bsd/	This is a port of the Debian operating system, complete with apt, dpkg, and GNU userland, to the NetBSD kernel	2001–2010
Debian port to HPPA (Debian-hppa)	http://lists.debian.org/debian-hppa/	This is a port to Hewlett-Packard's PA-RISC architecture	2001–2010
Debian port to Hurd (Debian-hurd)	http://lists.debian.org/debian-hurd/	The GNU Hurd is a totally new operating system being put together by the GNU group	1999–2010
Debian port to MIPS (Debian-mips)	http://lists.debian.org/debian-mips/	MIPS port of Debian GNU/Linux, able to run at both endiannesses	1999–2010
Debian port to PowerPC (Debian-ppc)	http://lists.debian.org/debian-powerpc/	PowerPC port of Debian GNU/Linux. The PowerPC architecture allows both 64-bit and 32-bit implementations	1999–2010
Debian port to SPARC (Debian-s390)	http://lists.debian.org/debian-s390/	Discussions on the IBM S/390 port of Debian GNU/Linux	2001–2010
Debian port to SPARC (Debian-sparc)	http://lists.debian.org/debian-sparc/	This port runs on the Sun SPARCstation series of workstations, as well as some of their successors in the sun4 architectures	1998–2010

Table 2
GA results for different values of coefficients $c1$, $c2$ and $c3$.

Parameters ($c1/c2/c3$)	Explained variance	Indicators	Factor number	Interpretability
1.00/0.00/0.00	77.73	20	7	0.00
0.80/0.10/0.10	79.80	49	12	0.00
0.60/0.20/0.20	78.03	46	10	0.30
0.40/0.30/0.30	73.87	53	10	0.18
0.20/0.40/0.40	75.70	49	10	0.30
0.00/0.50/0.50	72.40	48	9	0.44
0.00/0.00/1.00	59.65	14	2	1.00
0.10/0.10/0.80	*70.93*	*23*	*4*	*1.00*
0.20/0.20/0.60	71.60	36	5	0.60
0.30/0.30/0.40	74.94	49	10	0.30
0.40/0.40/0.20	77.28	51	11	0.27
0.50/0.50/0.00	75.16	52	11	0.08
0.00/1.00/0.00	72.53	55	11	0.09
0.10/0.80/0.10	72.52	54	11	0.18
0.20/0.60/0.20	75.37	54	12	0.17
0.30/0.40/0.30	76.79	53	12	0.16
0.40/0.20/0.40	75.31	43	8	0.50
0.50/0.00/0.50	73.24	22	5	0.80

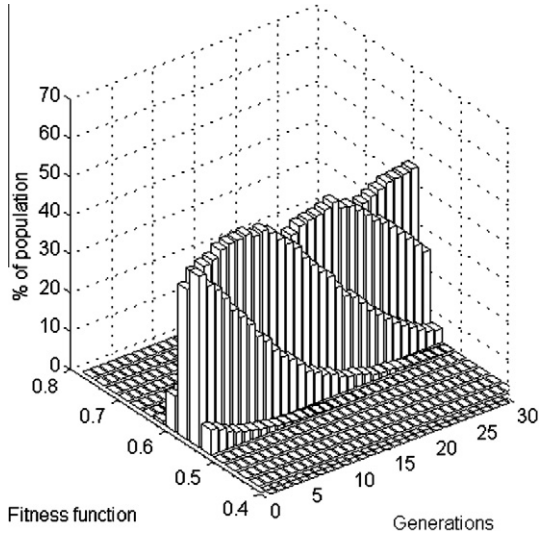

Fig. 7. Fitness distribution over 30 generations of the genetic algorithm.

The evolution of the genetic clustering algorithm is detailed in Fig. 7. The initial population (generation 0) has a low fitness value, which indicates that the individuals of the population are far from the optimum. As the number of generations increase, the fitness of individuals within the population also increases, as the genetic algorithm is biased towards the survival of genetic material contained within the individuals with high fitness function values.

The optimum subset of indicators provided by GA is listed in Table 3. In particular, the indicators description and the network over which they have been calculated are detailed.

The results from factor analysis using the set of variables selected by the genetic algorithm are detailed in Table 4. Usually, a number of factors equal to the number of eigenvalues higher than 1 is selected (Rencher, 2002). Consequently, up to four latent factors can be distinguished as result of factor analysis.

The indicators associated to each factor are obtained from the factor loadings using a Varimax rotation. All the indicators associated in this way with the same factor are hypothesized to share a common meaning that the analyst should discover. Table 5 shows which indicators are associated to each factor and their corresponding factor loadings.

On the other hand, factor scores are used to categorize the original sample of OSS communities, which can be approximated to one of the identified latent factors. Consequently, the original sample of OSS communities can be categorized in four groups. An analysis of variance (ANOVA) has been applied to the categorization of

Table 4
Explained variance of resulting factor analysis.

Factor	Eigenvalues		
	Value	Variance (%)	Cumulative (%)
1	*11.881*	*47.526*	*47.526*
2	*4.388*	*17.553*	*65.078*
3	*2.309*	*9.234*	*74.313*
4	*1.455*	*5.818*	*80.131*
5	0.914	3.658	83.789
6	0.799	3.195	86.984
7	0.652	2.606	89.590
⋮	⋮	⋮	⋮
23	0.000	0.001	100.000

Table 3
Selected set of indicators.

	Description	Network/subnetwork
VAR02	Number of interactions	Complete network
VAR03	Number of repeated interactions	Complete network
VAR04	Density	Complete network
VAR06	The Zagreb group index	Complete network
VAR09	Free riders	Complete network
VAR10	Active members	Complete network
VAR11	Members responsible of more than 50% of contributions	Complete network
VAR17	Normalized size of output-domain (standard deviation)	Complete network
VAR18	Proximity prestige (average value)	Complete network
VAR26	Network betweenness Centralization	Complete network
VAR29	Number of vertices with betweenness centrality >0	Complete network
VAR30	Egocentric density (average value)	Complete network
VAR31	Egocentric density (standard deviation)	Complete network
VAR33	Number of developed brokerage roles among active members	Complete network
VAR34	Number of vertices developing a brokerage role among active members and free riders	Complete network
VAR35	Number of developed brokerage roles among active members and free riders	Complete network
VAR41	Density	Active members network
VAR42	Average degree	Active members network
VAR45	Average out-degree	Active members network
VAR48	Normalized size of output-domain (average value)	Active members network
VAR50	Proximity prestige (average value)	Active members network
VAR53	Closeness centrality (average value)	Active members network
VAR59	Egocentric density (average value)	Active members network

Table 5
Identified factors.

		Description	Loading
F1	VAR02	Number of interactions	0.921
	VAR03	Number of repeated interactions	0.900
	VAR06	The Zagreb group index	0.836
	VAR09	Free riders	0.800
	VAR10	Active members	0.931
	VAR11	Members responsible of more than 50% of contributions	0.703
	VAR29	Number of vertices with betweenness centrality >0	0.945
	VAR33	Number of developed brokerage roles among active members	0.929
	VAR34	Number of vertices developing a brokerage role among active members and free riders	0.925
	VAR35	Number of developed brokerage roles among active members and free riders	0.910
F2	VAR17	Normalized size of output-domain	0.854
	VAR18	Proximity prestige (average value)	0.847
	VAR26	Network Betweenness Centralization	0.764
	VAR30	Egocentric density (average value)	0.881
	VAR31	Egocentric density (standard deviation)	0.719
F3	VAR48	Normalized size of output-domain (average value)	0.717
	VAR50	Proximity prestige (average value)	0.901
	VAR53	Closeness centrality (average value)	0.899
	VAR59	Egocentric density (average value)	0.740
F4	VAR04	Density (complete network)	0.705
	VAR41	Density (active members netw.)	0.897
	VAR42	Average degree	0.799
	VAR45	Average out-degree	0.703

Table 6
Statistical significance of ANOVA.

	F	Sig		*F*	Sig
VAR02	41.837	0.000	VAR31	16.644	0.000
VAR03	36.386	0.002	VAR33	41.910	0.000
VAR04	7237	0.003	VAR34	86.377	0.000
VAR06	24.322	0.000	VAR35	46.100	0.000
VAR09	50.931	0.000	VAR41	8018	0.000
VAR10	72.496	0.000	VAR42	10.950	0.000
VAR11	23.912	0.000	VAR45	6489	0.000
VAR17	53.598	0.000	VAR48	26.019	0.000
VAR18	40.664	0.000	VAR50	16.445	0.306
VAR26	31.039	0.000	VAR53	14.434	0.000
VAR29	73.973	0.000	VAR59	7488	0.000
VAR30	37.410	0.000			0.000

Table 7
Mean values of selected indicators.

	*F*1	*F*2	*F*3	*F*4
VAR02	11096.3846	2264.8140	1935.4688	1092.2500
VAR03	8102.5385	1553.1860	1464.1250	900.3333
VAR04	0.0201	0.0558	0.0268	0.0663
VAR06	1.9473E7	1.9473E7	1.9473E7	1.9473E7
VAR09	353.6154	353.6154	353.6154	353.6154
VAR10	377.9231	377.9231	377.9231	377.9231
VAR11	8.1538	8.1538	8.1538	8.1538
VAR17	0.2353	0.2353	0.2353	0.2353
VAR18	0.0894	0.0894	0.0894	0.0894
VAR26	0.1048	0.1048	0.1048	0.1048
VAR29	198.3462	198.3462	198.3462	198.3462
VAR30	0.2611	0.2611	0.2611	0.2611
VAR31	0.3163	0.3163	0.3163	0.3163
VAR33	60910.6538	60910.6538	60910.6538	60910.6538
VAR34	55.8846	55.8846	55.8846	55.8846
VAR35	4590.7692	4590.7692	4590.7692	4590.7692
VAR41	0.0742	0.0742	0.0742	0.0742
VAR42	383725.4850	383725.4850	383725.4850	383725.4850
VAR45	152903.9688	152903.9688	152903.9688	152903.9688
VAR48	0.7834	0.7834	0.7834	0.7834
VAR50	0.2969	0.2969	0.2969	0.2969
VAR53	0.2972	0.2972	0.2972	0.2972
VAR59	0.4730	0.4730	0.4730	0.4730

the original sample in the four groups obtained form factor analysis. The aim of this analysis consists of checking the null hypothesis of equal population means. Table 6 details the *F* statistic, the ratio of two different estimators of population variance, which appears together with its corresponding critical level or observed significance. The results is that the null hypotheses have been rejected in all the cases with a significance value below 0.05. That means the obtained categorization from factor analysis is well defined.

Table 7 details the mean value of the considered 23 indicators per each of the distinguished groups. Using this information as well as the factor loadings of Table 5, the following websites structure patterns can be distinguished:

Factor 1: This factor considers high sized communities characterized by a high number of interactions and a clear hierarchy among its members. The core group is located on top of this hierarchy, and they are responsible of developing a brokerage role among the rest of community members like active members and free riders. Factor 1 includes those communities with the highest number of members belonging to the core group.

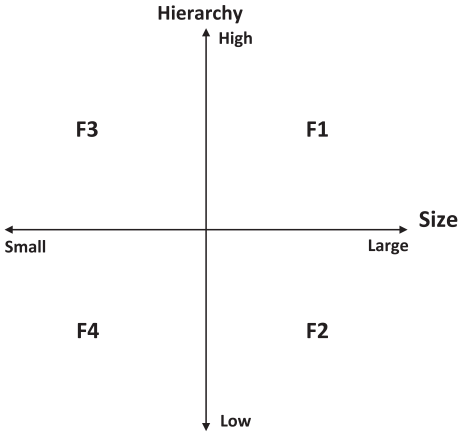

Fig. 8. Interpretation of identified factors.

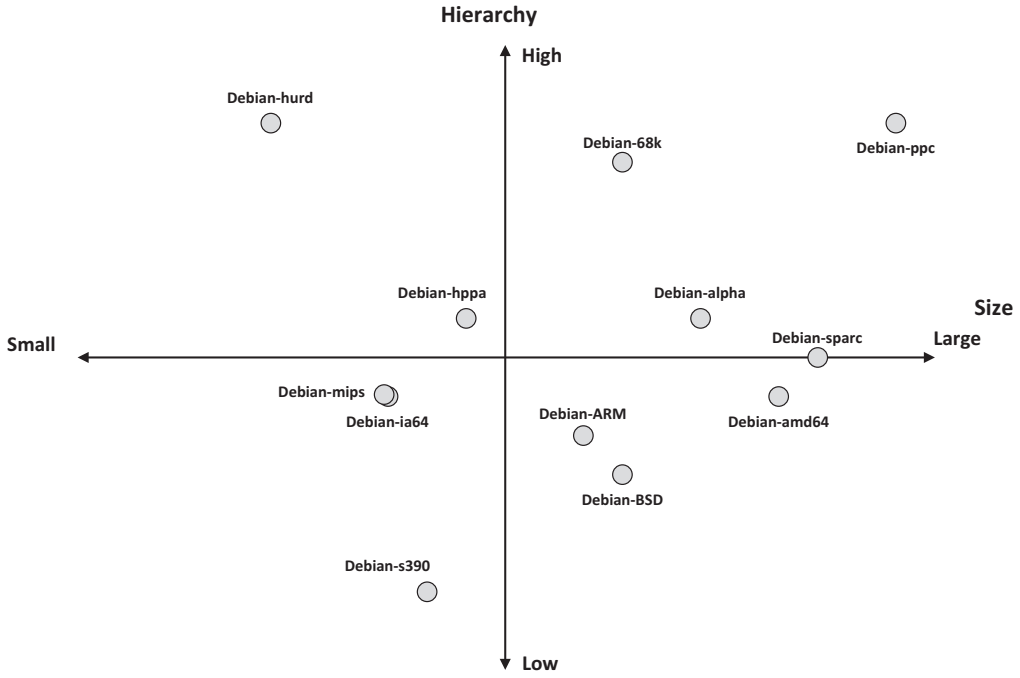

Fig. 9. Patterns of behaviour of Debian Linux ports communities.

Factor 2: Factor 2 refers to those communities with a high local connectivity. Communities following this pattern exhibit the highest average value of egocentric density and proximity prestige, which means their nodes are highly connected with their neighbours. Although these communities are not so high as those of Factor 1, they also exhibit an important size with a high number of interactions. However, the level of hierarchy is considerable lower. The role of the core group is shaded by the high activity of the rest of the community. In fact, F2 communities are the ones with the lowest number of free riders.

Factor 3: Factor 3 corresponds to small and centralized networks, where active members exhibit a high involvement and the core group also performs an important brokerage role. This factor includes communities with a strong hierarchy among their members but with a flatter structure compared to F1 communities.

Factor 4: Factor 4 considers the smallest communities but with a high density, which means that active members are highly interconnected. Despite of being smaller than F3 communities, they exhibit a higher average degree value. As a difference, the activity of the core group is shadowed by the active members of the community, so the hierarchy is much weaker than in F3 communities.

Obtained factors can be interpreted in terms of size and hierarchy of communities, as represented in Fig. 8. This figure graphically visualizes the four identified patterns of behaviour of OSS communities. Size represents how many users the OSS project is able to attract. Therefore, it can be interpreted as a measurement of how successful the underlying software is. Hierarchy refers to the internal organization of the community.

A high hierarchy means that the core group exerts an meaningful influence over the rest of the community while a low hierarchy means there is not a dominant group. Fig. 9 details the position of the considered Linux Debian communities in terms of size and hierarchy. The general trend is to be located on the right part of this figure as communities try to attract as many members as possible. As a difference, they follow different strategies regarding their internal hierarchy level. In general, they tend to maintain a certain hierarchy and only one community has a clear low hierarchy level.

One of the main implications of this results is that size and hierarchy are independent dimensions, so it is not incompatible a high size with a strong hierarchy. In fact, the growth in size impels the creation of hierarchical layers. In general, projects gain stability as more or less formal hierarchies emerge. Hierarchy mainly depends on the activity of the core group team. It is their responsibility to canalize the discussion, to provide solutions o alternatives to the posted problems and to facilitate the incorporation of qualified active members as part of this core group. Hierarchy also causes a rise in the entry barriers to the core group. Outsiders only can acquire authority inside the project through the legitimate peripheral participation procedure. Therefore, hierarchy guarantees that the project is under the control of the core group of developers.

6. Conclusions

This paper describes a procedure for the extraction of the patterns of behaviour of open source communities using a genetic search over a large set of Social Network Analysis indicators. As a difference to previous studies, only focused on small set of SNA features, the proposed evolutionary computation technique combined with factor analysis allows to consider all the possible metrics able to characterize interactions among community members. Obtained results show four patterns of behaviour that have been extracted as latent factors from the whole data set. These patterns can be in turn classified in terms of size and hierarchy of communities. Analyzed communities can be approached to a point in a bidimensional space described by these two dimensions. In general, OSS communities tend to gain as many members as possible, and their internal organization facilitates a certain hierarchy to emerge. Hierarchy can be understood as a stabilizing factor of the community that assigns the authority to a reduced core group member. Access to the inner circle of the community can only be achieved through a process of participation and interaction with the rest of the community.

Appendix A

Indicators	Description	Network
VAR01	Number of community members	Complete
VAR02	Number of interactions	Complete
VAR03	Number of repeated interactions	Complete
VAR04	Density	Complete
VAR05	Average degree	Complete
VAR06	The Zagreb group index	Complete
VAR07	The Randic connectivity index	Complete
VAR08	Index of relinking	Complete
VAR09	Free riders	Complete
VAR10	Active members	Complete
VAR11	Members responsible of more than 50% of contributions	Complete
VAR12	Average out-degree	Complete
VAR13	Standard deviation out-degree	Complete
VAR14	Number of components with more than 1 user	Complete
VAR15	Per cent of users included in components	Complete
VAR16	Normalized size of output-domain (average value)	Complete
VAR17	Normalized size of output-domain (standard deviation)	Complete
VAR18	Proximity prestige (average value)	Complete
VAR19	Proximity prestige (standard deviation)	Complete
VAR20	Maximum k-core (value of k)	Complete
VAR21	Number of users included in maximum k-core	Complete
VAR22	Number of k-cores with more than 1 user ($k > 0$)	Complete
VAR23	Per cent if users included in k-cores ($k > 0$)	Complete
VAR24	Closeness centrality (average value)	Complete
VAR25	Closeness centrality (standard deviation)	Complete
VAR26	Network Betweenness Centralization	Complete
VAR27	Average value of vertices betweeness centrality	Complete
VAR28	Standard deviation of vertices betweeness centrality	Complete
VAR29	Number of vertices with betweenness centrality > 0	Complete
VAR30	Egocentric density (average value)	Complete
VAR31	Egocentric density (standard deviation)	Complete
VAR32	Number of vertices developing a brokerage role among active members	Complete
VAR33	Number of developed brokerage roles among active members	Complete
VAR34	Number of vertices developing a brokerage role among active members and free riders	Complete
VAR35	Number of developed brokerage roles among active members and free riders	Complete
VAR36	Average brokerage role of core group with active members	Complete
VAR37	Average brokerage role of core group with free riders	Complete
VAR38	Average number of core group neighbours in the first level	Complete
VAR39	Average number of core group neighbours in all levels	Complete
VAR40	Average number of levels in which users are accessed by the core group	Complete
VAR41	Density	Active members
VAR42	Average degree	Active members
VAR43	The Randic connectivity index	Active members
VAR44	Index of relinking	Active members
VAR45	Average out-degree	Active members
VAR46	Standard deviation out-degree	Active members
VAR47	Per cent of users included in components	Active members
VAR48	Normalized size of output-domain (average value)	Active members
VAR49	Normalized size of output-domain (standard deviation)	Active members
VA050	Proximity prestige (average value)	Active members
VAR51	Proximity prestige (standard deviation)	Active members
VAR52	Per cent if users included in k-cores ($k > 0$)	Active members
VAR53	Closeness centrality (average value)	Active members
VAR54	Closeness centrality (standard deviation)	Active members
VAR55	Number of vertices with closeness centrality > 0	Active members
VAR56	Network betweenness centralization	Active members
VAR57	Average value of vertices betweeness centrality	Active members
VAR58	Standard deviation of vertices betweeness centrality	Active members
VAR59	Egocentric density (average value)	Active members
VAR60	Egocentric density (standard deviation)	Active members

References

Alander, J. T. (1992). On optimal population size of genetic algorithm. In *Proceedings CompEuro 1992. Computer systems and software engineering, 6th annual European computer conference* (pp 65–70).

Barcellini, F., Détienne, F., & Burkhardt, J.-M. (2009). Participation in online interaction spaces: Design-use mediation in an open source software community. *International Journal of Industrial Ergonomics, 39*(3), 533–540.

Crowston, K., & Howison, J. (2005). The social structure of free and open source software development. *First Monday, 10*(2).

Crowston, K., & Howison, J. (2006). Hierarchy and centralization in free and open source software team communications. *Knowledge, Technology, & Policy, 18*(4), 65–85.

De Jong, K. A. (1975). An analysis of behaviour of a class of genetic adaptive systems. Thesis. University of Michigan.

Deek, F. P., & McHugh, J. A. M. (2008). *Open source: Technology and policy*. NY: Cambridge University Press.

Devillers, J., & Balaban, A. T. (1999). *Topological indices and related descriptors in QSAR and QSPR*. The Netherlands: Gordon and Breach Science Publishers.

Durugbo, C. (2012). Modelling user participation in organisations as networks. *Expert Systems with Applications, 39*(10), 9230–9245.

Feczak, S., & Hossain, L. (2011). Exploring computer supported collaborative coordination through social networks. *The Journal of High Technology Management Research, 22*(2), 121–140.

Feller, J., & Fitzgerald, B. (2002). *Understanding open source software development*. London, UK: Addison-Wesley.

Freeman, L. C. (2004). *The development of social network analysis: A study in the sociology of science*. Vancouver, Canada: Empirical Press.

Gao, Y., & Madey, G. (2007). Network Analysis of the SourceForge.net Community. In J. Feller, B. Fitzgerald, W. Scacchi, A. Sillitti (Eds.), IFIP International Federation for Information Processing, vol. 234, Open Source Development, Adoption and Innovation, Boston, Springer, pp. 187–200.

Goldberg, D. E. (1989). *Genetic algorithm in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

Hinds, D. & Lee, R.M., 2008. Social network structure as a critical success condition for virtual communities. In *Proceedings of the 41st Hawaii international conference on system sciences* (pp. 323–333).

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press.

Hossain, L., Wu., A., & Chung, K. (2006). Actor centrality correlates to project based coordination. In: P. Hinds, D. Martin (Eds), *Proceedings of the CSCW'06 conference*, Banff, Canada (pp. 363–372).

Hossain, L., & Zhu, D. (2009). Social networks and coordination performance of distributed software development teams. *The Journal of High Technology Management Research, 20*(1), 52–61.

Kuk, G. (2006). Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List. *Management Science, 52*(7), 1031–1042.

Kwon, D., Oh, W., & Jeon, S. (2007). Broken Ties: The Impact of Organizational Restructuring on the Stability of Information-Processing Networks. *Journal of Management Information Systems, 24*(1), 201–231.

Lee, Y., & Lee, H. (2011). Application of factor analysis for service R&D classification: A case study on the Korean ICT industry. *Expert Systems with Applications, 3*(3), 2119–2124.

Long, Y., & Siau, K. (2007). Social network structures in open source software development teams. *Journal of Database Management, 18*(2), 25–40.

Martínez-Torres, M. R., & Toral-Marín, S. L. (2010). Strategic group identification using evolutionary computation. *Expert Systems with Applications, 37*(7), 4948–4954.

Mateos-Garcia, J., & Steinmueller, W. E. (2008). The institutions of open source software: Examining the Debian community. *Information Economics and Policy, 20*, 333–344.

Michalewicz, Z. (1996). *Genetic algorithm + data structures = evolution programs* (3rd ed.). Berlin, Germany: Springer-Verlag.

Mockus, A., Fielding, T., & Herbsleb, D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, 11*(3), 309–346.

Nooy, W., Mrvar, A., & Batagelj, V. (2005). *Exploratory network analysis with Pajek*. New York: Cambridge University Press.

Okoli, C., & Oh, W. (2007). Investigating recognition-based performance in an open content community: A social capital perspective. *Information & Management, 44*(3), 240–252.

Panchal, J. H. (2009), Co-evolution of products and communities in mass-collaborative product development – a computational exploration. In *Proceedings of international conference on engineering design (ICED'09)* (p. 147).

Pereira, C. S., & Soares, A. L. (2007). Improving the quality of collaboration requirements for information management through social networks analysis. *International Journal of Information Management, 27*(2), 86–103.

Preece, J. (2001). Sociability and usability in online communities: determining and measuring success. *Behaviour & Information Technology, 20*(5), 347–356.

Reina, D. G., Toral, S. L., Johnson, P., & Barrero, F. (2012). An evolutionary computation approach for designing mobile ad hoc networks. *Expert Systems with Applications, 39*(8), 6838–6845.

Robles, G., Gonzalez-Barahona, J. M., & Michlmayr, M. (2005). Evolution of volunteer participation in libre software projects: Evidence from Debian. In *Proceedings of the first international conference on open source systems, Genova* (pp. 100–107).

Rencher, A. C. (2002). *Methods of Multivariate Analysis. Wiley Series in Probability and Statistics* (2nd ed.). Berlin: Springer.

Singh, P. V. (2010). The small-world effect: The influence of macro-level properties of developer collaboration networks on open-source project success. *ACM Transactions on Software Engineering and Methodology, 20*(2), 1–27.

Sowe, S., Stamelos, I., & Angelis, L. (2006). Identifying knowledge brokers that yield software engineering knowledge in OSS projects. *Information and Software Technology, 48*(11), 1025–1033.

Toral, S. L., Martínez-Torres, M. R., & Barrero, F. (2009a). Virtual communities as a resource for the development of OSS projects: the case of Linux ports to embedded processors. *Behaviour and Information Technology, 28*(5), 405–419.

Toral, S. L., Martínez-Torres, M. R., Barrero, F., & Cortés, F. (2009b). An empirical study of the driving forces behind online communities. *Internet Research, 19*(4), 378–392.

Toral, S. L., & Martínez Torres, M. R. (2009). International comparison of R&D investment by European, US and Japanese Companies. *International Journal of Technology Management, 49*(1/2/3), 107–122.

Toral, S. L., Martínez Torres, M. R., & Barrero, F. (2010). Analysis of virtual communities supporting OSS projects using social network analysis. *Information and Software Technology, 52*(3), 296–303.

Valverde, S., Theraulaz, G., Gautrais, J., Fourcassie, V., & Sole, R. V. (2006). Self-organization patterns in wasp and open source communities. *IEEE Intelligent Systems, 21*(2), 36–40.

West, J., & O'mahony, S. (2008). The role of participation architecture in growing sponsored open source communities. *Industry & Innovation, 15*(2), 145–168.

Xu, J., Gao, Y., Christley, S. & Madey, G. (2005). A topological analysis of the open source software development community. In *Proceedings of the 38th annual Hawaii international conference on system sciences. HICSS '05* (pp. 188–198).

Xu, J., Christley, S., & Madey, G. (2006). Application of social network analysis to the study of open source software. In Jürgen Bitzer & Philipp J. H. Schröder (Eds.), *The economics of open source software development*. Elsevier Press.

Ye, Y., Nakakoji, K., et al. (2005). The co-evolution of systems and communities in free and open source software development. Free/open source software development. S. Koch. Hershey, PA, Idea Group Inc. (IGI) (pp. 59–82).