Who Contributes to What? Exploring Hidden Relationships between FLOSS Projects

M. Mahbubul Syeed, Imed Hammouda

Who Contributes to What? Exploring Hidden Relationships Between FLOSS Projects

M.M. Mahbubul Syeed[1] and Imed Hammouda[2]

[1] Department of Pervasive Computing
Tampere University of Technology, Finland
mm.syeed@tut.fi
[2] Department of Computer Science and Engineering
Chalmers and University of Gothenburg, Sweden
hammouda@chalmers.se

Abstract. In this paper we address the challenge of tracking resembling open source projects by exploiting the information of which developers contribute to which projects. To do this, we have performed a social network study to analyze data collected from the Ohloh repository. Our findings suggest that the more shared contributors two projects have, the more likely they resemble with respect to properties such as project application domain, programming language used and project size.

1 Introduction

With the exponential increase of Free/Libre Open Source (FLOSS) projects [6], searching for resembling open source components has become a real challenge for adopters [7]. By **resemblance**, we mean similarity factors between projects such as features offered, technology used, license scheme adopted, or simply being of comparable quality and size levels.

In this paper we exploit the information of 'which developers contribute to which FLOSS projects' to identify resembling projects. Our assumption is that if a developer contributes to several projects, simultaneously or at different times, then there might be implicit relationships between such projects. For example, one FLOSS project may use another as part of its solution [17] or two projects could be forks of a common base [16].

The idea of collecting and studying data of who contributes to which FLOSS projects is not new. The question has been the focus of many studies due to its relevance from many perspectives. For instance, the question has been significant for companies who want to identify who influences and controls the evolution of a specific project of interest[2], or to explore the social structure of FLOSS development [1], or simply to study what motivates people to join open source communities [3]. In this work, we address the following research questions:

1. How do FLOSS project development communities overlap?
2. To what extent can developer sharing in FLOSS projects approximate resemblance between the projects themselves?

For answering these questions, we used social network analysis techniques to analyze data collected from the Ohloh repository [4].

2 Study Design

This section presents in detail our study design, covering discussion on the data sources, required data sets, data acquisition, cleaning, and analysis process along with validation and verification of the analysis process.

2.1 Data Source

For this study we selected Ohloh data repository [4], which is a free, public directory of open source software projects and the respective contributors.

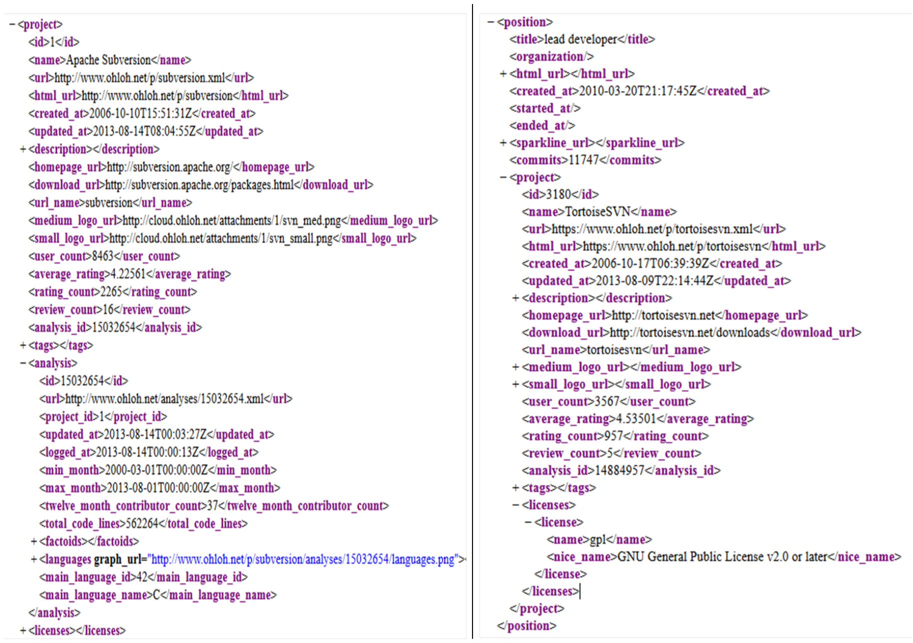
Ohloh collects and maintains development information of over 400 thousand FLOSS projects, and provide analysis of both the codes history and ongoing updates, and attributing those to specific contributors. It can also generate reports on the composition and activity of project code bases. These data can be accessed and downloaded through a set of open API which handles URL requests and responses [4]. The response data is expressed as an XML file, an example of which is shown in Fig. 1.

Our selection of Ohloh data repository is predominantly influenced by the following factors: (a) Ohloh data can be publicly reviewed, which in turn makes it one of the largest, most accurate, and up-to-date FLOSS software directories available; (b) the use of Ohloh repository makes FLOSS data available in a cleaned, unified and standard platform independent format. This makes the process of data analysis and visualization independent of technology, and data repository.

2.2 Data Collection

The following information has been collected from Ohloh repository in relation to this study:

Developer Account Information: An Account represents an Ohloh member, who is ideally a contributor to one or more FLOSS projects. Ohloh records a number of properties (or attributes) for an account.Among thousands of registered members, we collected account information of top 530 contributors according to assigned kudo rank of 10 and 9. Kudo rank is a way of appreciation to the FLOSS contributors through assigning a number between 1 and 10 for every Ohloh account [5].



```
− <project>
    <id>1</id>
    <name>Apache Subversion</name>
    <url>http://www.ohloh.net/p/subversion.xml</url>
    <html_url>http://www.ohloh.net/p/subversion</html_url>
    <created_at>2006-10-10T15:51:31Z</created_at>
    <updated_at>2013-08-14T08:04:55Z</updated_at>
  + <description></description>
    <homepage_url>http://subversion.apache.org/</homepage_url>
    <download_url>http://subversion.apache.org/packages.html</download_url>
    <url_name>subversion</url_name>
    <medium_logo_url>http://cloud.ohloh.net/attachments/1/svn_med.png</medium_logo_url>
    <small_logo_url>http://cloud.ohloh.net/attachments/1/svn_small.png</small_logo_url>
    <user_count>8463</user_count>
    <average_rating>4.22561</average_rating>
    <rating_count>2265</rating_count>
    <review_count>16</review_count>
    <analysis_id>15032654</analysis_id>
  + <tags></tags>
  − <analysis>
      <id>15032654</id>
      <url>http://www.ohloh.net/analyses/15032654.xml</url>
      <project_id>1</project_id>
      <updated_at>2013-08-14T00:03:27Z</updated_at>
      <logged_at>2013-08-14T00:00:13Z</logged_at>
      <min_month>2000-03-01T00:00:00Z</min_month>
      <max_month>2013-08-01T00:00:00Z</max_month>
      <twelve_month_contributor_count>37</twelve_month_contributor_count>
      <total_code_lines>562264</total_code_lines>
    + <factoids></factoids>
    + <languages graph_url="http://www.ohloh.net/p/subversion/analyses/15032654/languages.png">
      <main_language_id>42</main_language_id>
      <main_language_name>C</main_language_name>
    </analysis>
  + <licenses></licenses>
```

(a)

```
− <position>
    <title>lead developer</title>
    <organization/>
  + <html_url></html_url>
    <created_at>2010-03-20T21:17:45Z</created_at>
    <started_at/>
    <ended_at/>
  + <sparkline_url></sparkline_url>
    <commits>11747</commits>
  − <project>
      <id>3180</id>
      <name>TortoiseSVN</name>
      <url>https://www.ohloh.net/p/tortoisesvn.xml</url>
      <html_url>https://www.ohloh.net/p/tortoisesvn</html_url>
      <created_at>2006-10-17T06:39:39Z</created_at>
      <updated_at>2013-08-09T22:14:44Z</updated_at>
    + <description></description>
      <homepage_url>http://tortoisesvn.net</homepage_url>
      <download_url>http://tortoisesvn.net/downloads</download_url>
      <url_name>tortoisesvn</url_name>
    + <medium_logo_url></medium_logo_url>
    + <small_logo_url></small_logo_url>
      <user_count>3567</user_count>
      <average_rating>4.53501</average_rating>
      <rating_count>957</rating_count>
      <review_count>5</review_count>
      <analysis_id>14884957</analysis_id>
    + <tags></tags>
    − <licenses>
      − <license>
          <name>gpl</name>
          <nice_name>GNU General Public License v2.0 or later</nice_name>
        </license>
      </licenses>
    </project>
  </position>
```

(b)

Fig. 1. (a) Project Information (b) Developer Position Information

Project data: A Project represents a collection of source code, documentation, and web site data presented under a set of attributes, a list of which can be found in Fig. 1(a). We collected information of 4261 projects to which a total of 530 developers have contributed.

Position Information:A position is associated with each Ohloh account, which represents the contributions that the account holder has made to the project(s) within Ohloh. Information maintained in a position repository can be found in Fig. 1(b). We collected the position information for each of the 530 contributors.

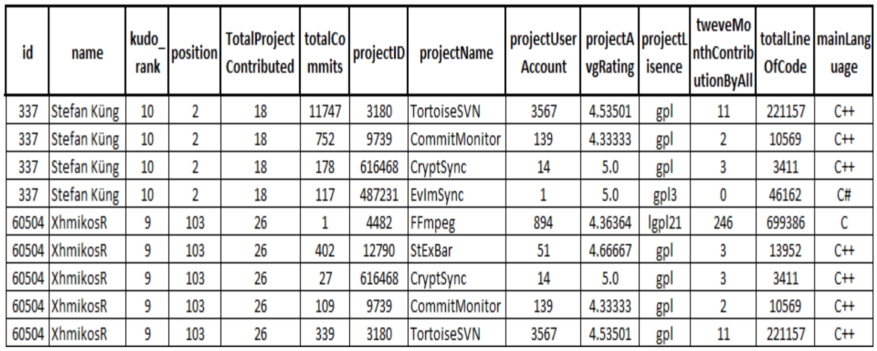
For downloading these repository data, we have implemented Java programs, one for each repository. Each Java program implements the API corresponding to a repository by combining the repository URL's and the unique API key to query the database. The result data set is in XML format.

2.3 Data Processing

From the collected repository data (i.e., the XML files as presented in Section 2.2), we parsed only the information that has significance to this study. The parsed information is recorded under a defined set of attributes/tags in XLSX files, one for each XML repository file.

Collected information is then merged to built a complete database required for data analysis. A partial snapshot of this database can be visualized in Fig. 2. Each row presents detailed information about (a) a contributor, (b) a project in which he/she contributed, and (c) the record of contribution to that project.

To automate data parsing and merging, parsers and data processors were written in Java. These programs use Jsoup HTML parser [9] and Apache POI [8] for parsing the XML files and to create database in XLSX format, respectively.



id	name	kudo_rank	position	TotalProjectContributed	totalCommits	projectID	projectName	projectUserAccount	projectAvgRating	projectLisence	tweveMonthContributionByAll	totalLineOfCode	mainLanguage
337	Stefan Küng	10	2	18	11747	3180	TortoiseSVN	3567	4.53501	gpl	11	221157	C++
337	Stefan Küng	10	2	18	752	9739	CommitMonitor	139	4.33333	gpl	2	10569	C++
337	Stefan Küng	10	2	18	178	616468	CryptSync	14	5.0	gpl	3	3411	C++
337	Stefan Küng	10	2	18	117	487231	EvlmSync	1	5.0	gpl3	0	46162	C#
60504	XhmikosR	9	103	26	1	4482	FFmpeg	894	4.36364	lgpl21	246	699386	C
60504	XhmikosR	9	103	26	402	12790	StExBar	51	4.66667	gpl	3	13952	C++
60504	XhmikosR	9	103	26	27	616468	CryptSync	14	5.0	gpl	3	3411	C++
60504	XhmikosR	9	103	26	109	9739	CommitMonitor	139	4.33333	gpl	2	10569	C++
60504	XhmikosR	9	103	26	339	3180	TortoiseSVN	3567	4.53501	gpl	11	221157	C++

Fig. 2. Partial Snapshot of the database

2.4 Data Analysis

Data analysis targeting to answer the research questions composed of two steps:

First, we created an **Implicit Network** in which two projects have a relationship if both are contributed to by the same contributor. An edge weight in this network represents the number of such common contributors between two projects. As an illustration, consider contributors *Stefan Küng* and *XhmikosR* in Fig. 2. The former contributor has contributed to 4 FLOSS projects as listed under the *projectName* column, while the latter has contributed to 5 projects. Both developers contributed to projects *TortoiseSVN*, *CryptSync*, and *CommitMonitor*. In total, Fig. 2 lists 6 distinct projects. An implicit network among these 6 projects is shown in Fig. 3. Projects *TortoiseSVN*, *CryptSync*, and *CommitMonitor* are linked with edge of weight 2 (i.e. 2 shared developers). All other edges have weight 1 as those project pairs have only one shared contributor. For example, *FFmpeg* and *CommitMonitor* have only developer *XhmikosR* in common.

The complete network is composed of 194424 edges between 4261 projects, with edge weight varies between 1 and 41. Complete edge list of this network can be found in [10].

Second, we measured the extent to which this implicit network comply with the factors often used to classify projects. For this study we selected five factors

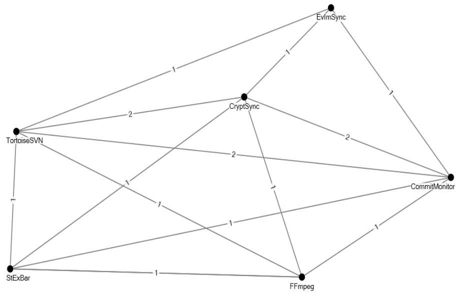


Fig. 3. An illustration of the implicit network

that are often cited by popular forges (e.g., SourceForge [11]) for categorizing FLOSS projects. These factors include programming language, project size, license, project rating [4] and project domain. Project size was further categorized into very large (500K SLOC), large (50K-500K SLOC), medium (5K-50K SLOC) and small (<5K SLOC), according to current literature [19]. Similarly, project rating was classified into top (≤ 4), high (≤ 3 and <4), medium (≤ 2 and <3) and low (<2) on a scale of 5.

For each factor, we identified from the implicit network the number of edges in which both projects have the same value. Due to the large size of the implicit network, we limited this investigation to top ranked 262 edges (the edges that have weight greater than 10) and least ranked edges (random selection of 500 edges from the edges that have weight of 1).

The result of this analysis is reported in Table 2. As an illustration of this approach, consider the project factor *Language* in Table 2. Among the top 262 edges, projects in 228 edges (87.03%) use same programming language, whereas among the bottom 495 edges, projects in 233 edges (47%) have same languages.

2.5 Program Verification

Two pass evaluations were conducted to verify the correctness of the implemented programs. First, the programs were tested with limited number of data samples taken from the collected data. Notified bugs (e.g., errors in parsed data for an HTML tag) were fixed accordingly. Second, a manual checking on a random sample of the actual collected data was done. The correctness of collected data in the second pass was reported to be over 98%.

3 Result Analysis

In this section we investigate the research questions primarily based on the implicit network (described in Section 2.4) revealing projects relationships based on common contributor(s), and by evaluating its compliance with the project factors (presented in Table 2) often used to classify FLOSS projects.

1. How do FLOSS project development communities overlap?
In this study we examined the implicit relationships among 2641 FLOSS projects that are contributed to by 530 contributors. Based on common contributor(s) as a relationship criterion, the implicit network reveals 194424 edges (implicit relations) between 4261 projects. Edge weight lays between 1 and 41, which simply reflects the total number of common contributors between projects. A partial snapshot of this network is shown in Figure 4, in which, for instance, projects *Debian* and *x.Org* have a relationship edge with weight 17. This is because 17 contributors contributed to both projects. This result, all together, portrays the collaborative nature of contribution by FLOSS community members.

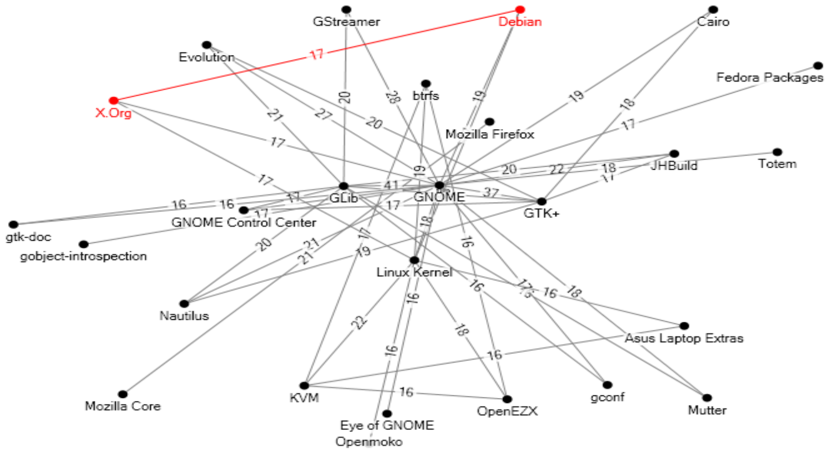


Fig. 4. Partial snapshot of the Implicit Network

To dig further and reason about such large deviation of edge weight, we counted the edges within a certain weight range, result of which is shown in Table 1. As presented in the table, the majority of the edges has low edge weight count (192559 edges out of 194424 have weight bellow 5). Investigating the cause, we observed that projects in these relationships are either medium or small size projects. Even in case where both projects have similar project sizes (3rd row of Table 2), 30% of the projects are medium or small sized. Hence, it is reasonable that communities of such projects should be small. Contrary to this, projects that are within the high edge weight count (e.g., edge weight over 10), are among the very large or large project groups, thus justifying the large overlapping community of contributors.

The above observation is analogous to the *richer gets rich* phenomenon in FLOSS projects collaboration [12], which states that communities that already had a high population would effectively attract more contributors.

Additionally, this structure of sharing contributors among multiple projects is supported by the small-world phenomenon [13]. In a small-world structure several projects are connected with each other through one or more links, e.g., common contributors. In this setup, with increasing number of common contributors, the communities of related projects (as realized by the implicit network) become strongly interconnected. We argue that this in turn may affect project success: The productivity of the contributors is boosted by providing them a dense communication channel to acquire more quantity and variety of information and knowledge resources [14].

Table 1. Edge count under edge weight category

Edge Weight Category	**Edge Count**
Less than 5	192559
Between 5 and 10	1603
Between 10 and 20	252
Greater than 20	10

Furthermore, contributors often participated in projects that belong to the same domain or are sub-projects to a larger one, and that utilizes same programming language(s) as development medium. High percentage of commonalities in implicit network under these two categories (as reported in row 6 and 2 of Table 2) vindicated the claim. This complies with the fact that contributors develop relationships on the common ground of interest [15].

> Based on the above observation and discussion it can be affirmed that FLOSS communities often prefer to participate in related projects with participation count varies with the size of the projects.

2. To what extent can developer sharing in FLOSS projects approximate resemblance between the projects themselves?

Within the scope of this investigation, we rationalized the projects relationship in implicit network against the actual factors that relate FLOSS projects. In doing so, we measured the extent to which the implicit network comply with the factors often used to classify projects. This approach is explained in detail in Section 2.4, and the result of which is presented in Table 2.

Among the five project factors, implicit network could effectively approximate three of them, namely, project domain, programming language and project

Table 2. Compliance of the edges in Implicit Network to that of project factors

Project Factor	**Edge Selection Category**	**Selected edges within the category**	**No of Edges in which both nodes have attribute value**	**Edges having same attribute value for the nodes**	**(%) count**	**Additional Info**
Language	Top [Edge weight >10]	262	262	228	87.03%	
	Bottom [Edge weight = 1]	500	495	233	47%	
Project Size	Top [Edge weight >10]	262	262	168	64.13%	Very large: 139 Large: 29
	Bottom [Edge weight = 1]	500	500	150	30%	Very large: 25 Large: 70 medium: 35 Small: 20
License	Top [Edge weight >10]	262	214	84	39.26%	
	Bottom [Edge weight = 1]	500	290	96	33.1%	
Project Rating	Top [Edge weight >10]	262	182	124	68.14%	Top: 107 High: 17
	Bottom [Edge weight = 1]	500	145	130	89.66%	Top: 120, High: 10
Project Domain	Top [Edge weight >10]	262	262	257	98%	
	Bottom [Edge weight = 1]	500	500	350	70%	

size. Column six in Table 2 shows high percentage of compliance of the implicit relationships to these project factors.

Contributors are most often attracted towards projects that fall within the same project domain or are the sub-projects of a larger one. As can be seen in row six of Table 2, among the top listed 262 edges, 257 (98%) has conformance to same project domain. Similar observation holds (with 70% of conformance) for bottom 500 edges as well. This implies that similar project domain most effectively creates favorable ground for attracting contributors to participate in them.

Language similarity is found to be one of the major selection factors for contributors participation. According to the data in the second row of Table 2, 87.03% of the top ranked 262 edges have language similarity in contrast to only 47% similarity for the bottom 500 edges. This observation approves that language similarity among projects offers strong support to attract large number of contributors.

The factor project size also imitate analogous results to that of language factor (row three in Table 2). Projects that are very large or large in size (64.13%) are able to manage larger collaborative contributor community than medium or smaller sized projects (30%).

Additionally, results on project rating show that contributors are more attracted towards highly rated projects than low rated ones (row 5 of Table 2).

However, contributors participation to the projects is not constrained by the licenses of the respective projects. Our investigation reported that very low percentage of projects in implicit network have same license terms (only 39.26%

of the top ranked edges and 33.1% of the bottom edges as presented in row 4 of Table 2).

> Projects that are linked in the implicit network with high edge weight count most likely belong to the same domain, use the same programming languages, or have similar project size.

The following aspects have been identified which could lead to threats to validity of this study.

External validity (how results can be generalized): This study includes 4261 FLOSS projects that are contributed by 530 contributors. Though, these projects cover a wide spectrum of FLOSS territory according to project size, domain, used languages and licenses, we cannot claim completeness of this justification.

Internal validity (confounding factors can influence the findings): The data used in this study is limited to the one provided by Ohloh and may raise trust concerns.

Construct validity (relationship between theory and observation): Data analysis programs written for this study produce data accuracy of over 98%, which was measured with random sample of collected data. This may affect the construct validity.

4 Conclusions

This paper studied to what extent resembling FLOSS components can be tracked based on community activities. Based on our findings, we claim that the proposed approach could approximate to a satisfactory degree resemblance between projects with respect to project domain, programming language and project size. Contrary to these factors, license terms of the projects came out as the least influential factor among all. This finding points out the fact that individual contributors may not be concerned about the licensing issues while selecting projects for contribution. These claims however, need further study. In this regard, a questionnaire to the open source community could be planned and carried out.

5 Acknowledgement

This work is funded by the Nokia Foundation Grant, 2013 and the TiSE Graduate school, Finland.

References

1. K. Crowston and J. Howison, "The social structure of Free and Open Source Software development" First Monday, 10(2), 2005.
2. T. Aaltonen and J. Jokinen: "Influence in the Linux Kernel Community", Proceedings of OSS 2007, pp. 203-208, Limerick, Ireland, June 2007.
3. A. Bonaccorsi and C. Rossi: "Altruistic individuals, selfish firms? the structure of motivation in open source software". First Monday (1-5) (2004).
4. Ohloh. http://www.ohloh.net/. Last accessed November 2013.
5. Ohloh kudo rank. http://meta.ohloh.net/kudos/. Last accessed Nov, 2013.
6. A. Deshpande and D. Riehle: "The Total Growth of Open Source", Proceedings of OSS 2008, pp. 197-209, Milan, Italy, September 2008.
7. J. Rudzki, K. Kiviluoma, T. Poikonen and I. Hammouda: "Evaluating Quality of Open Source Components for Reuse-Intensive Commercial Solutions", Proceedings of EUROMICRO-SEAA 2009, pp. 11-19, Aug. 2009.
8. Apache POI-Java API for Microsoft Documents. http: // poi.apache.org /. Last accessed September 2013.
9. jsoup: Java HTML Parser. http: // jsoup.org /. Last accessed September 2013.
10. Research Data, http://datasourceresearch.weebly.com/. Accessed in Nov. 2013.
11. Source Forge, sourceforge.net. Last accessed November 2013.
12. M. Weiss, G. Moroiu, P. Zhao, "'Evolution of Open Source Communities", Open Source Systems, IFIP International Federation for Information Processing Volume 203, pp. 21-32, 2006.
13. P. Vir singh, "The Small-World Effect: The Influence of Macro-Level Properties of Developer Collaboration Networks on Open-Source Project Success", ACM TOSEM, Vol. 20, No. 2, Article 6, 2010.
14. D. Watta, "'Networks, dynamics, and the small world phenomenon". Amer. J. Sociology 105, pp. 493527, 1999.
15. G. Madey, V. Freeh, R. Tynan, "The open source software development phenomenon: An analysis based on social network theory", In Americas conf. on Information Systems, pp. 1806-1813, 2002.
16. G. Robles and J.M. Gonzalez-Barahona,"A Comprehensive Study of Software Forks: Dates, Reasons and Outcomes", Open Source Systems, IFIP Volume 378, pp. 1-14, 2002.
17. H. Orsila, J. Geldenhuys, A. Ruokonen and I. Hammouda: "Update Propagation Practices in Highly Reusable Open Source Components", Open Source Systems, IFIP Volume 275, 2008.
18. J. M. Gonzalez-Barahona, G. Robles and S. Dueñas, "Collecting Data About FLOSS Development: The FLOSSMetrics Experience", Proceedings of FLOSS 2010, pp. 29-34, Cape Town, South Africa, 2010.
19. W. Scacchi, "Understanding Open Source Software Evolution: Applying, Breaking, and Rethinking the Laws of Software Evolution", John Wiley and Sons, 2003.