Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A multivariate classification of open source developers

Enrico di Bella[a], Alberto Sillitti[b,*], Giancarlo Succi[b]

[a] *Faculty of Economics, Università degli Studi di Genova, Genova, Italy*
[b] *Faculty of Computer Science, Free University of Bolzano, Bolzano, Italy*

A R T I C L E   I N F O

Article history:
Received 12 January 2011
Received in revised form 5 June 2012
Accepted 22 September 2012
Available online 5 October 2012

Keywords:
Open source
Development process
Empirical studies
Software metrics

A B S T R A C T

Open source software development is becoming always more relevant. Understanding the behavior of developers in open source software projects and identifying the kinds of their contributions is an essential step to improve the efficiency of the development process and to organize the development teams more effectively. Moreover, understanding the level of participation of the different developers helps to understand which members of the development team are more important than others and who are the actual key developers. This paper investigates the behavior of open source developers and the structure of the development of open source projects through the analysis of a very large dataset: 10 well-known and widely used open source software projects for a total of more than 4 MLOC (millions of lines of code) modified distributed in more than 200 K versions. This study builds on the top of other studies in this area applying a set of rigorous statistical techniques, analyzing how developers contribute to the projects. Its novelty is in the fine gain analysis of the developers that have commit rights on the repository of the project they work on, in the automated identification of key contributors of the project, in the size of the analyzed datasets, and in the statistical techniques used to classify the behavior of the developers in an automated way. To collect such large volume of data and to ensure their integrity, a tool to automatically mine open source version control systems has been used. The main result of this study is the identification of a recurrent pattern of four kinds of contributors with the same characteristics in all the projects analyzed even if the projects are very different in domain, size, language, etc.

1. Introduction

Open Source Software (OSS) projects are always more and more popular and their business relevance is significant. Consequently, there is a growing interest in such projects from a user perspective and also from the development model perspective [13].

Moreover, since many companies build their business on OSS (e.g., customizations, hardware and software products including OSS components, services, etc.), it is important for them to identify inside the community the key people that are the backbone of a OSS projects they base their business. This is important for several reasons including:

- The behavior of such key developers may have an impact on their business (e.g., leaving the project, forking the project, being hired by a competitor, etc.).

* Corresponding author. Address: Faculty of Computer Science, Libera Università di Bolzano, Piazza Domenicani 3, I-39100 Bolzano, Italy. Tel.: +39 0471 016134.

E-mail addresses: Enrico.diBella@economia.unige.it (E. di Bella), Alberto.Sillitti@unibz.it (A. Sillitti), Giancarlo.Succi@unibz.it (G. Succi).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.09.031

- Such developers are candidate for being supported by the company (e.g., hiring them as employees or consultants, sponsoring some activities, etc.).

There are a number of studies related to OSS and OSS development in particular. Most of these studies analyze the development process (e.g., [6]), the source code and its evolution (e.g., [22]), the skills, the motivation, and the role in the community of the developers, etc. (e.g., [5,42]). Even if some of these studies propose classifications of the people (developers and not) in open source communities (as listed in Section 2), in our knowledge none of them is related to the analysis of the developers that have commit rights to the Version Control System (VCS) of their project [16]. VCSs contains a huge set of information that can be exploited for extracting knowledge [32] and perform different kinds of empirical investigations [4,11,38,39]. In particular, our study goes beyond the existing ones analyzing the characteristics of such developers identifying recurrent patterns and key developers. Although there are important limitations (as described later), the committed code is the only measurable element that allows a classification of the developers from the point of view of development.

The aim of this paper is to analyze how developers contribute to a project and identify common characteristics and key developers. Such characteristics have to be general and not specific for a single project. To this end, the study considers ten medium-large projects belonging to different application domains (e.g., web browsers, application servers, window managers, etc.), written in different languages (C/C++ and Java), and with at least 15 developers able to commit the code into the VCS.

In most of the cases, it is difficult to evaluate the contribution of a single developer to a project. Usually, in an open source community, only a small subset of developers is able to commit code into the VCSs. Developers with no direct access have to rely on other developers that check the code, perform modifications, if needed, and commit the code [20]. According to Mockus et al. [19,20], committers are the most important developers in a project, therefore this study focuses on them. Since the goal of the paper is to identify how the main developers behave and not which are their specific contributions, it is not important for us if the code they commit is developed by them or reviewed and committed on behalf of other contributors.

We expect that even if the projects considered are very different, managed by different people, and with different rules, the behavior of the main developers can be clustered into a few categories common to most of the projects. For this reason, it is possible to identify which are the most important developers that a project cannot afford to lose and reorganize the development process to distribute responsibilities and code knowledge in a better way to improve the overall project (e.g., providing specific documentation or revising contribution rules).

Collected data support such hypothesis and the structure of the data allows the definition of an automated procedure to identify the different categories of developers in a project. In particular, there is a small group of *core developers* (contributing with most of the code and for an extended time) that are the key contributors for the project and the remaining developers can be classified into three well-defined groups according to their behavior with a decreasing importance for the project. These three groups are: *active developers* (contributing with a limited amount of code for an extended time), *occasional developers* (contributing with a limited amount of code for a short time), and *rare developers* (contributing with a very limited amount of code for a very limited time).

Existing research has indicated some general lines for the classification of the contributions of open source developers. Our research has progressed from the current research by (a) characterizing committers, (b) analyzing a much larger set of projects, and (c) using suitable statistical techniques, intended to analyze such large datasets. The result is a classification that is more likely to adhere to the reality, given the reduced uncertainty in the generalization. A side-result is the presentation of a general (and easily replicable) step by step multivariate statistical procedure that could be used also in other large-scaled empirical studies in software engineering.

The paper is organized as follows: Section 2 describes the related work; Section 3 presents the objectives of this work; Section 4 describes the data collection; Section 5 proposes a classification model for open source developers; Section 6 introduces the statistical techniques used; Section 7 validates the model against real data; Sections 8 and 9 discuss the results and identify the limitations of the study; finally, Section 10 draws the conclusions.

2. Related work

As stated in the previous section, there are several studies dealing with open source developers and most of them present analysis of their skills, motivation, role in the community, etc. [7,13].

In particular, there are studies related to the classification and the definition of roles of the people involved in open source communities including both users and developers. Such studies are mainly based on the two classifications proposed by Mokus et al. [19,20] and by Nakakoji et al. [21].

Mokus et al. [19,20] focus on the developers of the Apache web server and Mozilla and classify them in two categories: *core developers* and *non-core developers*. The former group includes the developers that have produced more than 80% of the total amount of code; the latter group includes all the others. In their study, all core developers have access to the VCS, while it is not stated how many of the non-core ones have access.

Nakakoji et al. [21] and Ye and Kishida [41] consider four projects: GNUWingnut, the Linux Support project, SRA-PostreSQL, and Jun. They consider the entire communities related to such projects (not just the developers) and define an onion-like structure in which they identify eight layers (Fig. 1).

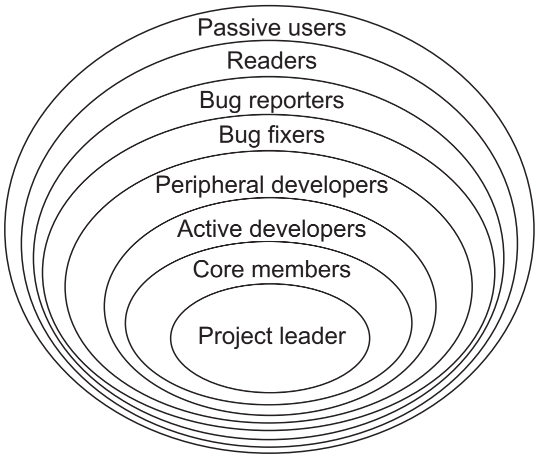

Fig. 1. Onion-like classification of the people in the community.

Developers that are contributing to the project with code are located in the five inner layers and it seems that only the project leader and the core members have write access to the VCS. This model refines the previous one analyzing deeply the role of the single person in the community and it is based on the kind of contribution (e.g., code, bug reports, code reviews, etc.) people provide.

If we exclude people that are not contributing providing source code, such classifications are mainly based on the following three parameters:

1. Write access into the VCS.
2. The amount of code developed.
3. The kind of contribution provided.

The first parameter is a Boolean value that is easily measurable, while the second and the third ones are more difficult to measure. This is true for both developers that are not able to commit the code directly and developers that are able to commit code.

The former group is not able to send directly the code to the VCS, therefore it is difficult to get information about the amount of code produced and the kind of contribution provided. Such information can be present in several places such as mailing lists, wikis, etc. However, it is not easy to find and analyze such data manually (because of the amount of information) and even more difficult in an automated way (because of the lack of structure in the data).

For the latter group the evaluation of the amount of code developed and the kind of contribution seams possible but is very difficult in practice because of two main problems:

1. Developers produce their own code but they also act as proxies for the former group revising and committing their code.
2. Usually, there are no explicit markers that identify the code developed by them and the code reviewed and committed for other developers.

As listed, the existence of different kinds of developers is well-known in the literature [42] but a few works define a rigorous and quantitative-based procedure to classify them in practice [5,6].

3. Objectives of the study

The study aims at characterizing the behavior of committers of OSS projects sine their behavior could have an impact on businesses relying on such products. We are not investigating the specific impact such behaviors have on the life of the project or on the business based on such products but just find out if all the committers behave in the same way in the same project and if there are identifiable behavioral clusters. Moreover, we would like to find out if such clusters are present in very different projects (different domains, size, etc.).

In our knowledge, there are no studies investigating the behavior of the developers that are able to commit code and define the characteristics of such developers identifying recurrent patterns in their behavior and key developers.

This paper is a contribution in this area defining a set of statistical techniques to evaluate in a rigorous way the behavior of developers in projects. This approach has been adopted to evaluate very different open source projects. Through this investigation, we have find out that there are four categories of developers with specific characteristics in all the projects we have considered. The developers belonging to such categories have a different impact on the project they are working

Table 1
Main papers in the area.

Paper	Main results
Feller and Fitzgerald [7] Goldman and Gabriel [13]	Analysis of developers skills, motivation, role in the community, etc.
Mokus et al. [19,20]	Analysis of the Apache web server and Mozilla. Classification of developers in two categories: *core developers* and *non-core developers*. Core developers produce more than 80% of the total amount of code
Nakakoji et al. [21] and Ye and Kishida [41]	Analysis of the entire community (not just developers) of four projects and the definition of the onion-like structure

on and the departure of people from such different categories has very different potential impact on the evolution of the project (see Table 1).

Framed in terms of research questions, we aim at presenting evidence that will allow us to reject (or accept) the following null hypotheses:

- H_{A0}: there are no differences in the behavior of the committers of an open source project.
- H_{B0}: different projects do not show any common behavioral patterns of the committers.

4. Data collection

The study has been carried out through the analysis of 10 projects (Table 2) written in C/C++ and Java. We have considered very different projects in terms of domain, size, language, management, etc. but all of them are community-based. This because we would like to provide a more general characterization that can be applied to almost any project. Moreover, in the analysis, we have considered only parameters that can be extracted automatically and present in all possible projects (i.e., extracted from the Version Control System). Other parameters (including qualitative ones) could provide a more reliable representation of the projects but they have not been considered, since the aim of the work is to define a fully automated methodology [23,40,24,2,36,14,37,3,10].

Data have been extracted from the main branches of the VCS. Some of the projects considered include a small amount of code written with different programming languages (i.e., Ant, Perl, Javascript, and Bash scripts); however, such contributions are small and should not affect the results of the analysis since the amount of code is less than 5% of the entire project. In this study, we do not consider the potential effects of such extra code that should be investigated in the future. We also do not considered the files written in XML, HTML, text, etc.

Table 2 shows the basic characteristics of the projects included in the study:

- The number of unique contributors that can commit code into the repositories. Usually, these developers do not only develop code but they also collect code from other developers, evaluate it, and accept or reject the contributions [7]. The number of developers that are actually contributing to the projects is much higher. However, it is not possible to determine such number from the data stored in the version control repositories.
- Languages used.
- ΔLOC: number of lines of code (LOC) added since the beginning of the project. It is calculated as follows:

$$\sum_{i=1}^{N}(\Delta\mathrm{LOC})_i$$

where $(\Delta\mathrm{LOC})_i = \mathrm{LOC}_{i,end} - \mathrm{LOC}_{i,start}$ and N is the number of files in the project written in C, C++, and Java.

Table 2
List of the open source projects analyzed.

Project	Developers	Language	ΔLOC	KLOC	Versions
Ant	29	Java	475,238	230	21,667
Apache 1.2	16	C/C++	49,194	500	2031
Avalon	30	Java	390,465	130	10,793
Cocoon 2.1	40	Java	302,160	1200	7922
Jakarta Hivemind	46	Java	449,074	130	10,759
Kde	379	C/C++	2,319,569	15,800	76,273
Mozilla Firefox	24	C/C++	625,159	4,300	13,536
Mozilla Thunderbird	182	C/C++	1,390,218	740	36,593
XML Xalan	31	C/C++/Java	768,569	540	19,288
Xfree86-4	19	C/C++	1,274,638	3200	12,518

- KLOC: number of thousands of lines of code of the project at the end of the data collection. The number includes just the lines of code written in C, C++, and Java.
- Number of versions of all the files in the project. It is calculates as follows:

$$\sum_{i=1}^{N} v_i$$

where v_i is the number of versions of the file i and N is the number of files in the project written in C, C++, and Java.

The absolute values of LOC provided in Table 2 are provide just to give an idea of the size of the projects but it has not used in the analysis. This is because some projects started from an empty repository (e.g., Xalan), while others started importing a previous version (e.g., Apache). This does not represent a problem for the analysis, since the study focuses on the evolution of the source code, therefore only the variations are considered and showed in the following analysis.

5. Our classification model

In this work, we use three statistical tools to explore the latent structure of the programming activity and to classify developers in a project [25–30]: Principal Components Analysis [17], Factor Analysis [31,15,1], and Cluster Analysis [35,9,8].

Cluster Analysis includes a collection of techniques that are used to group multidimensional entities according to various criteria of their degree of homogeneity and heterogeneity. The problem of clustering N data vectors in p-space into k clusters can be handled in many ways and the most appropriate technique to use depends upon the problem. In this work, we use a Non Hierarchical Clustering (k-means clustering) which is based upon the criterion of minimization of the variance within the clusters and which can be thought as an ANOVA (ANalysis Of VAriance) *in reverse* as the methodology moves objects (e.g., cases) in and out of groups (clusters) to get the most significant ANOVA results. Usually, as the result of a k-means Clustering Analysis, we would examine the means for each cluster on each dimension to assess how distinct our k clusters are. Ideally, we would obtain very different means for most, if not all, dimensions used in the analysis.

We propose a classification model focused only on the developers that are able to commit into the VCS based on:

1. The observation of the behavior of the developers.
2. The models described in the subsequent section.

To validate the technique, we consider ten well-known open source projects and the data available from their code repositories. The proposed classification is:

1. *Core developers*, who develop most of the code and contribute to the project for an extended period of time.
2. *Active developers*, who develop limited part of the project for an extended period of time.
3. *Occasional developers*, who provide a limited contribution in a limited timeframe.
4. *Rare developers*, who provide a very limited contribution during very short periods of the life of the project.

We have identified such categories as a result of the analysis we performed on two of the considered projects (Apache 1.2 and Mozilla Firefox). The three-dimensional plot of the properties we discuss in the next section produces four well-defined clusters of developers as shown in Fig. 2, this allows us to define the four categories.

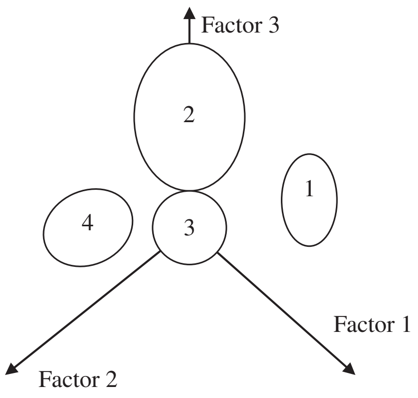

Fig. 2. Typical structure of developers in a project: 1 = core developers, 2 = active developers, 3 = occasional developers, and 4 = rare developers.

Table 3
List of the variables available for the analysis.

Variable code	Variable name	Variable description
DiP	Days in the project	Number of days the developer has been in the project (time between first and last commit)
NoC	Number of commits	Number of atomic modifications made to the source code at file level made by the developer
IT	Inter-commit time	Average time (days) between commits
NF	Number of files	Number of files the developer edited
LOC	Lines of code	Number of lines of code added or removed by the developer since the beginning of the project
C	Complexity	McCabe cyclomatic complexity [18] added or removed by the developer since the beginning of the project
SM	Structural modifications	Any changes to the source code that affect the structure of the product done by the developer. Such modifications include: class and function definitions, decision statements, loops, etc.
NSM	Non structural modifications	Any changes to the source code that do not affect the structure of the product done by the developer. Such modifications include: variable definitions, inclusions, assignments, etc.
CM	Comment modifications	Any changes to the comments done by the developer

6. Statistical techniques

The variables available for the analysis are nine (Table 3) for all the developers involved in the ten well-known open source projects considered (Table 2) and the data are extracted from the related VCS. Because of the high dimensionality of the problem, the statistical methodology we use is a "three-phase" Multivariate Analysis (Principal Components, Factor Analysis, and Cluster Analysis) which has been implemented separately for each project.

As it is becoming important to develop automatic procedures for the analysis of VCS [12,33,34,42], we have defined a common procedure for all the projects analyzed that is easily repeatable for any other project in an automated way.

Even if there are a number of variables that could be split into two subvariables (e.g., "Lines of code" into "Added lines of code" and "Removed lines of code") we decided not to have such a distinction because it does not produce significant improvements to the model.

As data represents populations and not samples (because we analyze all the data for the considered projects) the results are not estimates but advanced descriptive statistics and any inference on the subsequent results is meaningless.

7. Empirical validation of the model

To highlight the latent structure of developers' behavior to classify the developers into the groups defined in Section 4, we use a Principal Components Analysis on the variables shown in Table 3 to select the most important ones in terms of eigenvalues. As these components are linear combinations of the original variables and contain most of the total variability of the developers' behavior, they can be used in place of the complete set of variables for the classification purpose.

To interpret better their meaning, the components are rotated in the attempt to produce eigenvectors with values close to −1, 0, or 1. There are different kinds of rotations that can be applied but the choice of the best rotation of the factors is linked to the specific dataset available [31]. The possible choices belong to two different groups: orthogonal and oblique rotations.

In this analysis, we have considered the most common normalized and raw orthogonal rotations (varimax, quartimax, biquartimax, and equamax) and we have decided to use a biquartimax normalized rotation of factors since it produces the best (in terms of interpretation of factors) and most stable results across the various projects considered. The results produced by this rotation are reported in Tables 4–6.

Table 4
Factor loadings (biquartimax normalized) for the first factor.

Factor 1	Ant	Apache	Avalon	Cocoon	JH	Kde	FFox	TB	Xalan	Xfree
DiP	0.83	0.40	0.46	0.35	0.19	0.37	0.30	0.36	0.40	0.46
NoC	0.82	0.60	0.83	0.58	0.39	0.97	0.93	0.89	0.67	1.00
IT	−0.17	−0.19	−0.16	−0.13	−0.16	−0.11	−0.23	−0.12	−0.15	−0.20
NF	0.75	0.13	0.72	0.23	0.25	0.74	0.83	0.45	0.25	1.00
LOC	0.86	0.96	0.95	0.26	0.69	0.99	1.00	0.97	0.87	1.00
C	0.98	0.99	0.99	0.97	0.97	0.99	0.92	0.99	0.97	1.00
SM	0.99	0.97	0.99	0.96	0.96	0.99	0.98	0.99	0.95	1.00
NSM	0.99	0.98	0.99	0.94	0.96	0.99	0.99	0.99	0.92	1.00
CM	0.98	0.87	0.99	0.94	0.81	0.96	0.97	0.98	0.95	1.00
EV[a]	6.57	5.14	6.27	4.22	4.19	6.46	6.44	5.97	5.06	7.19
PT (%)[b]	73.0	57.1	69.6	46.9	46.5	71.8	71.6	66.4	56.2	79.9

[a] EV: explained variance.
[b] PT: percentage of the total variance explained by the given principal component.

Table 5
Factor loadings (biquartimax normalized) for the second factor.

Factor 2	Ant	Apache	Avalon	Cocoon	JH	Kde	FFox	TB	Xalan	Xfree
DiP	0.22	0.81	−0.11	0.16	0.77	−0.01	−0.11	−0.14	0.20	−0.18
NoC	0.03	0.73	0.17	0.80	0.84	0.04	0.03	0.10	0.74	0.05
IT	−0.98	−0.26	−0.97	−0.12	−0.20	−0.99	−0.96	−0.97	−0.16	−0.97
NF	0.14	0.92	0.24	0.96	0.92	0.13	0.15	0.28	0.95	0.05
LOC	0.05	0.26	0.08	0.95	0.58	0.02	0.04	0.06	0.53	0.05
C	−0.01	0.04	0.00	0.18	0.07	0.01	0.04	0.02	0.11	0.05
SM	0.00	0.12	0.01	0.19	0.26	0.01	0.05	0.03	0.13	0.05
NSM	0.02	0.16	0.02	0.30	0.24	0.02	0.04	0.02	0.34	0.05
CM	0.04	0.27	0.00	0.16	0.46	0.03	0.06	0.01	0.18	0.05
EV[a]	1.02	2.29	1.04	2.68	2.86	1.00	0.98	1.06	1.98	1.00
PT (%)[b]	11.4	25.5	11.6	29.8	31.8	11.2	10.9	11.8	22.0	11.1

[a] EV: explained variance.
[b] PT: percentage of the total variance explained by the given principal component.

Table 6
Factor loadings (biquartimax normalized) for the third factor.

Factor 3	Ant	Apache	Avalon	Cocoon	JH	Kde	FFox	TB	Xalan	Xfree
DiP	0.07	0.07	0.80	0.75	0.09	0.90	0.94	0.83	0.71	0.87
NoC	−0.56	0.05	0.40	0.13	0.04	0.15	0.06	0.33	0.24	0.08
IT	0.05	−0.95	0.03	−0.88	−0.97	0.00	0.10	−0.03	−0.88	0.13
NF	−0.60	0.19	0.47	0.14	0.10	0.40	0.24	0.73	0.23	0.08
LOC	−0.44	0.06	0.23	0.11	0.08	0.07	0.05	0.20	0.15	0.08
C	0.03	0.06	0.01	0.13	0.06	0.02	0.02	0.11	0.19	0.08
SM	0.03	0.06	0.04	0.15	0.04	0.04	0.03	0.12	0.20	0.08
NSM	−0.01	0.06	0.05	0.12	0.05	0.05	0.05	0.12	0.17	0.08
CM	−0.06	0.04	−0.02	0.15	0.07	0.08	0.07	0.07	0.16	0.08
EV[a]	0.88	0.96	1.08	1.47	0.97	1.01	0.96	1.41	1.55	0.81
PT (%)[b]	9.8	10.6	12.0	16.3	10.8	11.2	10.7	15.7	17.2	9.0

[a] EV: explained variance.
[b] PT: percentage of the total variance explained by the given principal component.

According to the analysis performed, the first component is by far the most important (mean 63.9%, median 68% of total variance) and it is mainly correlated with the variables LOC, C, SM, NSM, and CM and it can be named "Programming activity". The other two components are equally relevant–the eigenvalues are similar and close to one; their composition is homogeneous for the different projects but the ordering is not always the same, according to the eigenvalues. Even if they switch their role as second or third component, the correlations they have with the other variables are quite similar.

The interpretation of the second and third factors is not straightforward. The former is mainly connected with the IT variable; therefore, we name the factor "Inter-commit time". The latter is connected with the variables NF and DiP, therefore we name the factor "Extent of the contribution".

Fig. 3 shows the structure of factor loadings for the KDE project (even if most of the labels are nor readable, it is clear that the values lay on the Factor 1 except NF, DiP, and IT). Such a structure, very similar for all the projects analyzed, shows that

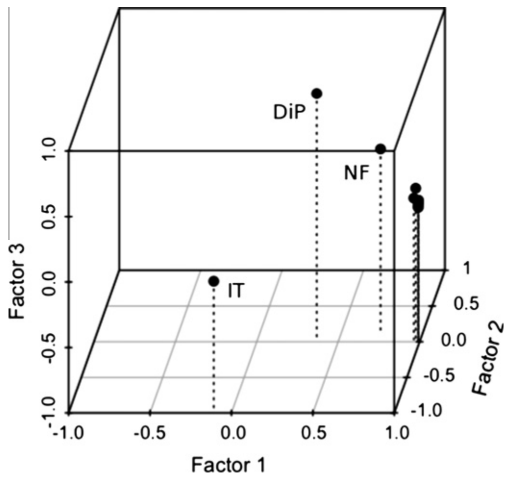

Fig. 3. Factor loadings for KDE, Factor 1 vs. Factor 2 vs. Factor 3. Extraction: Principal Components, rotation: biquartimax normalized.

the variables LOC, C, SM, NSM, and CM are pretty well identified with the Factor 1, the variable IT with the Factor 2, and the remaining variables DIP and NF are correlated with all the three factors. This situation can be extended to the other projects.

If the factors are rotated by an oblique transformation, the rotated factors become correlated. Often, oblique rotations produce more useful patterns than orthogonal rotations. However, a consequence of correlated factors is that there is no single unambiguous measure of the importance of a factor in explaining a variable. Therefore, we decided not to consider

Table 7
Factor structures for some oblique rotations for the KDE project.

Factor	Rotation quartimin			Rotation obvarimax			Rotation obquartimax			Rotation obequamax		
	F1	F2	F3	F1	F2	F3	F1	F2	F3	F1	F2	F3
DiP	0.43	0.56	0.09	0.22	0.93	−0.10	0.43	0.97	0.02	0.17	0.92	−0.14
NoC	0.95	0.86	−0.37	0.92	0.71	−0.41	0.97	0.52	−0.17	0.88	0.73	−0.52
IT	−0.11	−0.06	0.29	0.02	0.00	0.95	−0.11	0.01	0.99	0.10	0.03	0.90
NF	0.71	0.91	−0.35	0.64	0.77	−0.42	0.77	0.66	−0.23	0.59	0.78	−0.49
LOC	0.99	0.75	−0.29	0.96	0.66	−0.40	0.99	0.45	−0.14	0.93	0.69	−0.51
C	1.00	0.69	−0.27	0.96	0.62	−0.38	0.99	0.41	−0.13	0.94	0.65	−0.49
SM	1.00	0.71	−0.27	0.96	0.63	−0.39	0.99	0.42	−0.14	0.94	0.66	−0.50
NSM	1.00	0.73	−0.28	0.96	0.64	−0.39	0.99	0.44	−0.14	0.94	0.67	−0.50
CM	0.95	0.73	−0.28	0.92	0.64	−0.39	0.96	0.45	−0.15	0.90	0.67	−0.50

Table 8
Mean values of the variables for the four groups.

	Ant	Apache	Avalon	Cocoon	JH	Kde	FFox	TB	Xalan	Xfree
Group 1										
DiP	1664	381	773	494	722	1676	266	1321	869	1304
NoC	4483	206	2002	1928	667	5088	313	2247	1628	12381
IT	0	2	0	0	1	0	1	1	1	0
NF	770	47	843	802	233	638	40	244	309	5259
LOC	100622	6687	119074	57015	57382	162628	9588	90724	101436	1266848
C	4158	671	2401	2499	2257	9802	583	6446	3346	77962
SM	12858	1088	9770	6386	7852	23707	1541	15685	9928	165715
NSM	21385	2307	20471	17282	13510	44553	3270	27587	23069	392409
CM	9183	734	8349	4152	4430	16149	1065	12609	22134	131754
N	2	4	1	1	2	6	1	4	3	1
Group 2										
DiP	630	462	689	344	995	1261	72	917	718	1147
NoC	3971	175	716	1726	1034	367	10	231	1957	28
IT	0	3	3	0	1	19	12	17	0	100
NF	913	63	317	1720	466	121	4	101	857	21
LOC	63939	2580	21956	110893	32921	9533	556	6456	59948	962
C	740	142	309	19	279	469	61	388	653	50
SM	2264	284	1381	29	1736	1257	124	1004	2270	172
NSM	5056	747	2744	2111	3175	2406	210	1709	8715	286
CM	3044	274	1021	39	1510	1030	66	552	4934	104
N	2	6	12	1	5	99	16	44	5	2
Group 3										
DiP	414	142	89	280	289	254	375	185	469	101
NoC	207	31	16	133	137	46	13	43	220	8
IT	5	3	13	10	5	17	45	11	10	9
NF	103	18	11	88	64	17	8	22	98	7
LOC	6350	1393	646	4193	5457	1955	524	1358	7834	611
C	202	117	20	157	128	104	42	94	335	49
SM	599	197	55	477	507	272	87	228	1008	93
NSM	1319	394	126	1055	882	521	188	372	1906	201
CM	656	167	54	310	536	276	91	149	1721	95
N	23	5	12	32	31	203	3	101	21	8
Group 4										
DiP	1	1	152	20	115	223	58	211	1	220
NoC	1	1	1	1	2	2	2	3	1	2
IT	101	101	157	94	97	149	104	115	101	128
NF	1	1	1	1	2	2	1	3	1	2
LOC	29	1	32	11	69	46	7	31	3	122
C	2	0	1	0	2	2	0	2	0	1
SM	3	0	5	1	9	5	2	7	0	4
NSM	12	0	6	3	14	12	1	9	1	15
CM	2	0	1	0	8	7	0	6	0	1
N	2	1	5	6	8	71	4	33	2	8

oblique rotations (even if they produced good results) because they cannot be implemented easily in an automated analysis system and require some manual tuning for each project under analysis.

To evaluate the effect of an oblique rotation on factors, Table 7 shows the factor structures of some oblique rotations for the KDE project. Since oblique factors are correlated, it is difficult to assign a meaning to each factor as different factors can be correlated to the same variables.

Using the first three components (which represent approximately the 93–94% of total variability) as a basis for a non-hierarchical Cluster Analysis, it is possible to state that four groups are an optimal clustering for the developers. In particular, the developers in each project can be divided into four groups or categories:

- *Group 1 (Core developers):* a very small group of persons giving the most important contributions.
- *Group 2 (Active developers):* developers who are big contributors in a lot of different files and do a large part of the job.
- *Group 3 (Occasional developers):* a big number of developers contributing occasionally and focusing on specific files with a limited number of contributions. They contribute to the development of specific features or bug fixes.
- *Group 4 (Rare developers):* a relatively small number of persons who initially are very interested in the project but after a while they stop their contribution.

Proofs of the distinction among the clusters that this approach has identified are:

- the different means of the groups (a statistical test cannot be performed as data are populations and not samples),
- the results are very similar in all the projects analyzed (Table 8).

Table 9
Relative importance of all the categories for some important variables.

	Group 1 (%)	Group 2 (%)	Group 3 (%)	Group 4 (%)	Group 1 (%)	Group 2 (%)	Group 3 (%)	Group 4 (%)
	Number of programmers				*Lines of code*			
Ant	6.90	6.90	79.30	6.90	58.90	37.40	3.70	0.00
Apache	25.00	37.50	31.30	6.30	62.70	24.20	13.10	0.00
Avalon	3.30	40.00	40.00	16.70	84.00	15.50	0.50	0.00
Cocoon	2.50	2.50	80.00	15.00	33.10	64.40	2.40	0.00
JH	4.30	10.90	67.40	17.40	59.90	34.40	5.70	0.10
Kde	1.60	26.10	53.60	18.70	93.40	5.50	1.10	0.00
FFox	4.20	66.70	12.50	16.70	89.80	5.20	4.90	0.10
TB	2.20	24.20	55.50	18.10	92.00	6.50	1.40	0.00
Xalan	9.70	16.10	67.70	6.50	59.90	35.40	4.60	0.00
Xfree	5.30	10.50	42.10	42.10	99.90	0.10	0.00	0.00
Average	6.50	24.10	52.90	16.40	73.40	22.90	3.70	0.00
Median	4.30	20.20	54.50	16.70	73.40	19.80	3.10	0.00
	Complexity				*Structural modifications*			
Ant	81.50	14.50	4.00	0.00	81.80	14.40	3.80	0.00
Apache	72.20	15.30	12.60	0.00	69.30	18.10	12.60	0.00
Avalon	87.90	11.30	0.70	0.00	87.10	12.30	0.50	0.00
Cocoon	93.40	0.70	5.90	0.00	92.60	0.40	6.90	0.00
JH	84.70	10.50	4.80	0.10	77.70	17.20	5.00	0.10
Kde	94.50	4.50	1.00	0.00	93.90	5.00	1.10	0.00
FFox	85.00	8.90	6.10	0.00	87.90	7.10	5.00	0.10
TB	93.00	5.60	1.40	0.00	92.70	5.90	1.30	0.00
Xalan	77.20	15.10	7.70	0.00	75.20	17.20	7.60	0.00
Xfree	99.90	0.10	0.10	0.00	99.80	0.10	0.10	0.00
Average	86.90	8.60	4.40	0.00	85.80	9.80	4.40	0.00
Median	86.50	9.70	4.40	0.00	87.50	9.70	4.40	0.00
	Non structural modifications				*Comment modifications*			
Ant	77.00	18.20	4.70	0.00	71.30	23.60	5.10	0.00
Apache	66.90	21.70	11.40	0.00	62.50	23.30	14.20	0.00
Avalon	87.70	11.80	0.50	0.00	88.60	10.80	0.60	0.00
Cocoon	84.50	10.30	5.20	0.00	92.20	0.90	6.90	0.00
JH	76.80	18.10	5.00	0.10	68.30	23.30	8.30	0.10
Kde	93.80	5.10	1.10	0.00	92.50	5.90	1.60	0.00
FFox	89.10	5.70	5.10	0.00	87.20	5.40	7.40	0.00
TB	93.00	5.80	1.30	0.00	94.70	4.10	1.10	0.00
Xalan	68.50	25.90	5.70	0.00	76.90	17.10	6.00	0.00
Xfree	99.90	0.10	0.10	0.00	99.80	0.10	0.10	0.00
Average	83.70	12.20	4.00	0.00	83.40	11.50	5.10	0.00
Median	86.10	11.00	4.90	0.00	87.90	8.40	5.50	0.00

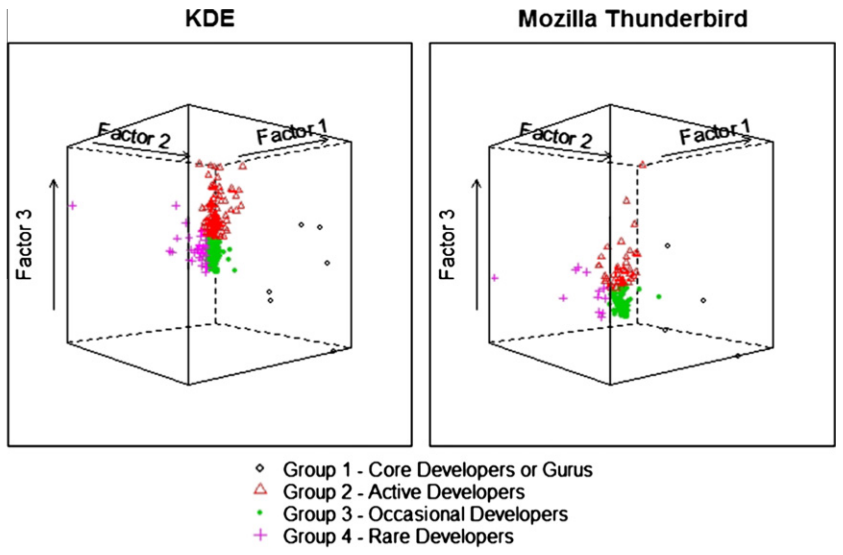

Fig. 4. Clusters of developers for KDE and Thunderbird.

Although not presented in this work, we obtained similar results also using a hierarchical clustering procedure using a complete linkage as amalgamation measure and Squared Euclidean Distances.

The relative importance of any group of developers for some relevant variables is show in Table 9. We can point out that 4.3% (as a median value since it is a more robust value than the average) of developers (core developers) produce 73.4% of the LOC and 86–87% of C, SM, NSM, and CM.

Fig. 4 shows some three-dimensional representations of the biggest projects (according to the number of developers). The three axes represent the first three components and the four types of dots represent the different groups of developers as defined through the clustering procedure. The different colors identify the different groups and it is clear that there are clusters. We have three factors, therefore we have a 3D picture. Very similar graphs can be obtained for all the projects analyzed and that justifies the structure proposed in Fig. 2.

8. Discussion

The analysis performed allow us to reject both H_{A0} and H_{B0} since we are able to identify four groups of developers able to commit code inside different open source communities. Such groups are well defined and characterized by the three factors identified through the Principal Components Analysis that are able to explain 93–94% of the total variability of the data:

- *Factor 1 – Programming activity:* mainly correlated to LOC, C, and the number of modifications to the code.
- *Factor 2 – Inter-commit time:* mainly correlated to IT.
- *Factor 3 – Extent of the contribution:* mainly correlated to NF and DiP.

Moreover, such groups are present in all the ten projects considered with similar characteristics even if they are very different from the point of views of the domain and the development community.

According to the characteristics of the identified groups, the people belonging to the different groups have different roles in the project:

- *Core developers:* such developers are a very small group but they are the backbone of the project taking the most important decisions about the architecture and the evolution of the project. Even if the present study is not investigating the effect of the departure from the project of such kinds of developers, we expect that the departure of people of this group can cause relevant problems, the creation of a fork, or the death of the project since they have most of the knowledge and they are the main contributors.
- *Active developers:* such developers have a good knowledge about the project and they are candidate to become *core developers*. They are involved in the key decisions of the project but they role is in most of the cases marginal. Even if the present study is not investigating the effect of the departure from the project of such kinds of developers, we expect that the departure of people of this group can produce some delays to the project but the overall impact is limited since most of the knowledge about the project is still present and just minor aspect miss. However, they can be recovered without producing major problems.
- *Occasional developers:* most of the developers belongs to this group focusing on specific aspects of the project (enhancements or bug fixes in specific module or subsystems). They are mainly involved in the development of specific modules or

subsystems but they do not have a project-wide role. Even if the present study is not investigating the motivation for such developers to contribute of the effect of the departure from the project of such kinds of developers, we expect that developers in this group are interested in the project because it is an important component for a commercial or another open source product that relies on the features they contribute. Moreover, we expect that the departure of people of this group do not affect the overall project but may have some (limited) impact on the specific modules or subsystems they contribute.

- *Rare developers:* the developers belonging to this group has obtained the commit rights to the project but they did not provide very relevant contributions and they stop their work for several reasons (e.g., they are not really interested, they realize the actual effort required for an active participation, etc.). Even if the present study is not investigating the effect of the departure from the project of such kinds of developers, we expect that the departure of people of this group do not affect the project in any way.

Monitoring the contributors of the different kinds of developers helps in understanding the health of a project, the potential evolution, and if investing in it (in terms of time or money) is a potential good or a bad investment.

This analysis is the first step towards an accurate analysis of the relationship among developers, for instance following the social networks approach proposed by Lopez-Fernandez et al. (2000) but focusing the attention only on some clusters of developers. For instance, considering that the KDE project has 372 developers, the potential number of one-to-one relationships is:

$$\binom{372}{2} = 69006$$

If we exclude from the analysis the group of rare developers, the number is consistently reduced to 45,150. Moreover, the study of the relationships among core developers can be done considering only 15 connections. The statistical approach to developers' clustering can be used to reduce the complexity of social networks analysis for software projects as it represent a step to prune developers with limited contributions to the project.

9. Limitations of the study

This work presents a number of limitations. The projects considered are ten open source projects selected among the well-known ones; therefore, it is not possible to generalize our conclusions to all open source projects or to proprietary projects.

The main limitation for the classification of the developers is the collection of a complete set of information. Is that it is very difficult to connect VCS repositories information with other types of information (e.g., newsgroups and forums) because while VCS repositories are exhaustive for one type of information (i.e., the variables specified in Table 3) not all the communications among developers pass through forums or can be traced. Moreover, the present study is not investigating the effect of the departure from the project the different kinds developers, therefore only some guesses are reported in the paper. A more detailed analysis of such aspects is required.

10. Conclusions

Understanding of developers' behavior is the starting point for analyzing the efficiency of a software project (both open and close source). Even if project leaders usually have a clear perception of the contribution of the single developers, this is rarely convertible into a quantitative measure and the transmission of such knowledge is even more difficult. In this work, we use a well-defined and easily repeatable statistical procedure, which can be used to cluster developers. However, this clustering in meaningful only inside a single project and the extracted values cannot be used to compare projects.

The study outlines that whatever the size of the considered projects, it is possible to define clearly four groups of developers grouped according to how they contribute to the project and the departure of developers is not always associated to problems for a project since it can produce a very marginal effect. Monitoring the contributors helps in understanding the health of a project, the potential evolution, and if investing in it (in terms of time or money) is a potential good or a bad investment.

Since one of our goals for this work is to propose a model, completely repeatable and based on common data available for all projects, we use standard analysis techniques that can be used on any project regardless the specificity of the data available. However, further work is required to improve the quality of the results tuning automatically the statistical techniques on the base of the specificity of the data available in each project.

Moreover, further work is required to verify if the approach works also in the case of proprietary software development in which factors such as non-volunteer participation to projects may influence the behavior of developers and, in particular, we expect a minor relevance of the Groups 3 (Occasional developers) and 4 (Rare developers).

The proposed procedure is completely automated, it has been replicated for all the projects and it is available to other researchers for replication.

Acknowledgement

We would like to thank Prof. Witold Pedrycz for his valuable comments to an early version of this paper.

References

[1] R.B. Cattell, The Scientific Use of Factor Analysis in Behavioural and Life Sciences, Plenum Press, 1978.
[2] J. Clark, C. Clarke, S. De Panfilis, G. Granatella, P. Predonzani, A. Sillitti, G. Succi, T. Vernazza, Selecting components in large COTS repositories, Journal of Systems and Software 73 (2) (2004).
[3] I. Coman, A. Sillitti, Automated identification of tasks in development sessions, in: 16th IEEE International Conference on Program Comprehension (ICPC 2008), Amsterdam, The Netherlands, 2008, pp. 10–13.
[4] L. Corral, A. Sillitti, G. Succi, J. Strumpflohner, J. Vlasenko, DroidSense: a mobile tool to analyze software development processes by measuring team proximity, in: 50th International Conference on Objects, Models, Components, Patterns (TOOLS Europe 2012), Prague, Czech Republic, 2012, pp. 29–31.
[5] K. Crowston, J. Howison, The social structure of free and open source software development, First Monday 10 (2) (2005).
[6] K. Crowston, Q. Li, K. Wei, U.Y. Eseryel, J. Howison, Self-organization of teams for free/libre open source software development, Information and Software Technology 49 (6) (2007).
[7] J. Feller, B. Fitzgerald, Understanding Open Source Software Development, Pearson, 2002.
[8] V.A. Fernández, M.D. Jiménez-Gamero, B. Lagos-Álvarez, Divergence statistics for testing uniform association in cross-classifications, Information Sciences 180 (23) (2010) 4557–4571.
[9] D. Fisch, B. Kühbeck, B. Sick, S.J. Ovaska, So near and yet so far: new insight into properties of some well-known classifier paradigms, Information Sciences 180 (18) (2010) 3381–3401.
[10] I. Fronza, A. Sillitti, G. Succi, An interpretation of the results of the analysis of pair programming during novices integration in a team, in: 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009), Lake Buena Vista, FL, USA, October 2009, pp. 15–16.
[11] I. Fronza, A. Sillitti, G. Succi, J. Vlasenko, M. Terho, Failure prediction based on log files using random indexing and support vector machines. Journal of Systems and Software, in press, http://dx.doi.org/10.1016/j.jss.2012.06.025.
[12] D. German, A. Mockus, Automating the measurement of open source projects, in: 3rd Workshop on Open Source Software Engineering, Portland, Oregon, 2003.
[13] R. Goldman, R.P. Gabriel, Innovation Happens Elsewhere: Open Source as Business Strategy, Morgan Kaufmann, 2005.
[14] A. Janes, M. Scotto, W. Pedrycz, B. Russo, M. Stefanovic, G. Succi, Identification of defect-prone classes in telecommunication software systems using design metrics, Information Sciences 176 (24) (2006).
[15] I.T. Joliffe, Principal Component Analysis, Springer-Verlag, 1986.
[16] L. Lopez-Fernandez, G. Robles-Martinez, J.M. Gonzalez-Barahona, Applying Social Network Analysis to the Information in CVS Repositories, 2004.
[17] K.V. Mardia, J.T. Kent, J.M. Bibby, Multivariate Analysis, Academic Press, 1979.
[18] T. McCabe, A complexity measure, IEEE Transaction on Software Engineering 2 (4) (1976).
[19] A. Mokus, R.T. Fielding, J. Herbsleb, A case study of open source development: the apache server, in: 22nd International Conference on Software Engineering, Limerick, Ireland, May 2000.
[20] A. Mockus, R.T. Fielding, J. Herbsleb, Two case studies of open source software development: Apache and Mozilla, ACM Transactions on Software Engineering and Methodology 11 (3) (2002).
[21] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, Y. Ye, Evolution patterns of open-source software systems and communities, in: International Workshop on Principles of Software Evolution (IWPSE 2002), Orlando, FL, USA, 2002.
[22] J.W. Paulson, G. Succi, A. Eberlein, An empirical study of open source and closed source software products, IEEE Transactions on Software Engineering 30 (4) (2004).
[23] W. Pedrycz, G. Succi, P. Musı̈lek, X. Bai, Using self-organizing maps to analyze object-oriented software measures, Journal of Systems and Software 59 (1) (2001).
[24] W. Pedrycz, G. Succi, M.G. Chun, Association analysis of software measures, International Journal of Software Engineering and Knowledge Engineering 12 (3) (2002).
[25] W. Pedrycz, K. Hirota, Forming consensus in the networks of knowledge, Engineering Applications of Artificial Intelligence 20 (5) (2007) 657–666.
[26] W. Pedrycz, P. Rai, A multifaceted perspective at data analysis: a study in collaborative intelligent agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B 38 (4) (2008) 1062–1072.
[27] W. Pedrycz, P. Rai, Experience-consistent modeling: regression and classification problems, Automatica 45 (2) (2009) 449–455.
[28] W. Pedrycz, From logic descriptors to granular logic descriptors: a study in allocation of information granularity. Journal of Ambient Intelligence and Humanized Computing (2012).
[29] W. Pedrycz, Allocation of information granularity in optimization and decision-making models: towards building the foundations of Granular Computing, European Journal of Operational Research, in press, http://dx.doi.org/10.1016/j.ejor.2012.03.038.
[30] W. Pedrycz, M. Song, A genetic reduction of feature space in the design of fuzzy models, Applied Soft Computing 12 (9) (2012) 2801–2816.
[31] S.J. Press, Applied Multivariate Analysis, Dover Publications, 1972.
[32] A. Rezaei, B. Rossi, A. Sillitti, G. Succi, Knowledge extraction from events flows, in: G. Anastasi, E. Bellini, E. Di Nitto, C. Ghezzi, L. Tanca, E. Zimeo (Eds.), Methodologies and Technologies for Networked Enterprises, Springer, 2012.
[33] G. Robles-Martinez, J.M., Gonzalez-Barahona, J. Centeno-Gonzalez, V. Matellan-Olivera, L. Rodero-Merino, Studying the evolution of libre software projects using publicly available data, in: 3rd Workshop on Open Source Software Engineering, Portland, OR, USA, 2003.
[34] G. Robles-Martinez, S. Koch, J.M. González-Barahona, Remote analysis and measurement of libre software systems by means of the CVSAnalY tool, in: Workshop on Remote Analysis and Measurement of Software Systems, Edinburgh, UK, 2004.
[35] H.C. Romesburg, Cluster Analysis for Researchers, Lulu.com., 2004.
[36] M. Scotto, A. Sillitti, G.Vernazza, T. Succi, A relational approach to software metrics, in: ACM Symposium on Applied Computing (SAC 2004), 2004.
[37] M. Scotto, A. Sillitti, G. Succi, T. Vernazza, A non-invasive approach to product metrics collection, Journal of Systems Architecture 52 (11) (2006).
[38] A. Sillitti, G. Succi, J. Vlasenko, Toward a better understanding of tool usage, in: 33th International Conference on Software Engineering (ICSE 2011), Honolulu, HI, USA, May 2011, pp. 21–28.
[39] A. Sillitti, G. Succi, J. Vlasenko, Understanding the impact of pair programming on developers attention: a case study on a large industrial experimentation, in: 34th International Conference on Software Engineering (ICSE 2012), Zurich, Switzerland, 2012, pp. 2–9 (June).
[40] G. Succi, L. Benedicenti, T. Vernazza, Analysis of the effects of software reuse on customer satisfaction in an RPG environment, IEEE Transactions on Software Engineering 27 (5) (2001).
[41] Y. Ye, K. Kishida, Toward an understanding of the motivation open source software developers, in: 25th International Conference on Software Engineering (ICSE 2003), Portland, OR, USA, 2003.
[42] L. Yu, S. Ramaswamy, Mining CVS repositories to understand open-source project developer roles, in: 4th International Workshop on Mining Software Repositories (MSR'07), Minneapolis, MN, USA, 2007.