Who are Source Code Contributors and How do they Change?

Massimiliano Di Penta
Dept of Engineering, University of Sannio
Benevento, Italy
dipenta@unisannio.it

Daniel M. German
Dept. of Computer Science, University of Victoria
Victoria, Canada
dmg@uvic.ca

***Abstract*—Determining who are the copyright owners of a software system is important as they are the individuals and organizations that license the software to its users, and ultimately the legal entities that can enforce its licensing terms, and change its license. In this paper we describe the difficulties of identifying the explicit copyright owners of a system, and those who contribute source code to it–who could potentially claim are also copyright owners of it.**

The paper introduces a method to track the names of contributors, including those explicitly listed as copyright owners from licensing statements in source code file. Then, it reports an empirical study performed on four open source systems—namely ArgoUML, Mozilla, Samba, and Squid—aimed at investigating the characteristics of their contributors and how they relate to the commits recorded in the system and users who perform them (its committers).

Results indicate that explicit contributors and copyright owners are not necessarily the most frequent committers. Also, they are often added during larger changes than average.

***Keywords*-Mining software repositories; open source systems; source code ownership; empirical study.**

I. Introduction

Intellectual Property (IP) clearance is becoming a major problem for organizations wanting to reuse Free/Open Source Software (FOSS). This problem has prompted an industry dedicated to address it. For example, BlackDuck http://www.blackducksoftware.com/ offers a *Software Intellectual Property Assessment Service* to businesses wanting to reuse FOSS, either internally or in products they sell.

A major question in IP clearance is: *who owns the source code of a specific FOSS project?* The answer to this question has important consequences: only the owners of the source code are capable of enforcing its licensing terms (*i.e.*, suing somebody for inappropriately using or reusing their source code) and are the only ones capable of licensing it [1]. Any organization or individual who is interested in incorporating a FOSS or commercial product into their own project should be aware of who its owner is, as it is only from the owner that a license can be acquired, and only the owner can initiate a copyright infringement lawsuit.

In the so-called proprietary software world (*i.e.*, non-FOSS) companies routinely negotiate software licensing contracts with its owner. In contrast, in the FOSS world, software is made available under one or more licenses by its owners, and the recipient should accept such licensing terms before reusing this software [2], [3] (see [4] for a description of how licensing affects reuse of software). Sometimes, however, the recipient is not willing (or cannot) accept the terms of a license. For example, an organization might want to include a library licensed under the General Public License inside a program that is to be sold in binary form only, but do not want to make its source code available. In such cases, this organization has two (legal) choices: either do not use the library, or negotiate a different license with the owner of the library [5]. For example, anybody wanting to reuse MySQL as part of a non-FOSS product must negotiate a commercial license with its owner (Sun Microsystems) [6].

The answer to the question "Who owns the source code of a specific Free/Open Source Software (FOSS) project?" is not simple. Is it the person (or persons) who mostly contributed to a software project? Is it their employer? Is it the person who authored the majority of the lines of code of the system as of today? Is the person with the largest number of commits? What if the system is a "fork" of another project, developed by a different group of individuals? [1] To the best of our knowledge, the existing software engineering literature has not investigated this problem yet. Existing works limit the analysis to versioning system committers [7], [8], [9].

The contribution of this paper is two-fold. First, it proposes a method to mine the owners of a software system. Second, it reports an empirical study, performed on four FOSS projects—ArgoUML, Mozilla, Samba, and Squid—showing that ownership in FOSS is not a trivial issue, and exploring how contributor names appear in source code files, and in what context they are introduced.

This paper is organized as follows. After Section II provides a short background discussion on software intellectual property, Section III illustrates how contributors are mentioned in source code files, while Section IV describes our method to extract contributor-related information from source code files. Section V describes the empirical study we performed, which results are reported and discussed in Section VI. Section VII discusses the related literature, and, finally, Section VIII concludes the paper and outlines directions for future work.

II. Background

Almost everywhere in the world, software is protected by copyright legislation [10]. While copyright laws vary from country to country, they are based on common guidelines set by the World Intellectual Property Organization (WIPO).

In general, any code fragment is protected by copyright when it is created. Very often, a source code file owner is its author; however when a developer is being employed by an organization to create such code, such an organization becomes its owner. In legal terms, the owner of the code is the owner of its copyright. The copyright owner of source code is the only one allowed to make copies of it, and to create derivative works from it. The copyright owner can transfer these rights to another party, and this is done using a license. For instance, if A is the copyright owner of code C, and B wants to include it as part of a product D, then B must obtain a license for C that allows it to use it inside D. FOSS licenses are primarily designed (under certain conditions) to permit reuse [3]. For example, if A licenses C under the terms of the General Public License version 3, then B can incorporate C inside D as long as D is licensed under the terms of the General Public License version 2 too (one of the main conditions imposed by the license).

From a legal point of view, copyright can be sold and transferred like any other type of property. Some organizations maintain strict ownership of their FOSS projects. For example, the Free Software Foundation (FSF) requires a transfer of ownership (*i.e.*, a transfer of copyright) for any contribution to Emacs (*i.e.*, the author of the contribution must sign a legal document transferring the copyright of the contribution—or patch—to the FSF). The same is done by Sun Microsystems—the current owner of MySQL, acquired from its previous owner, MySQL AB, in 2008—who requires that any contribution to MySQL should include a copyright transfer of the contribution to Sun Microsystems [11]. Even though both organizations have very different goals (one is striving to make FOSS more widely available, while the other is a commercial entity whose ultimate goal is to create revenue for its stock holders) they both want to be the unique owners of their products. In this way they are both capable of suing anybody who is inappropriately using it; Sun also offers commercial licenses to its database system (regardless of the wishes of any of the contributors who have transferred their copyright to them, and without paying them any portion of the royalties[1]).

Most FOSS projects, however, do not have such policies, and could have from one owner (projects developed by one developer only) to literally hundreds (as in the case of the Linux and the *BSD kernels). The copyright ownership in FOSS is made more ambiguous by the collaborative nature of its software development process. Frequently products are co-developed, over a long period, by many individuals, some of them regular maintainers, and others who submit sporadic patches (see [1] for an extensive analysis of the issues surrounding IP ownership).

[1]This is similar to the manner in which scientific conference proceeding and journal publishers request a transfer of copyright from the author. In this way, the publisher owns the copyright of the papers and can initiate legal action against anybody illegally making copies of them. Otherwise, the publisher would have to request each author to sue the infringer.

III. Who are the copyright owners of a FOSS?

In FOSS projects, source code files are the basic unit being licensed. As a consequence, each file usually contains the licensing terms under which it is made available to others. For example, the Free Software Foundation recommends that each source code file should include, at its beginning, a license notice, such as *"This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version."*

FOSS proponents (such as the Debian Project, the Free Software Foundation, and the Apache Foundation) and licenses (such as the BSD, the various GNU Licenses, the Apache License) recommend that every source code file contains a copyright ownership statement, that includes the year of the copyright, and the name of the author(s) that claim such copyright. Throughout this paper, we will refer to this area of the file as its *license statement* and will usually contain a description of the file, a copyright ownership statement, and its licensing terms.

When a file is copied from one system to another, is licensing statement is expected to be preserved (*e.g.*, the BSD license makes this a condition of granting a license). In previous research we have observed that this is indeed true [12]. We discovered that files have been copied from FreeBSD to Linux and vice-versa, preserving their licensing terms and copyright owners. FOSS developers appear concerned with properly following the conditions of licenses, and crediting copyright to its rightful owners.

The Debian Project, the publisher of the Debian Distribution (which is the basis of several other distributions including Ubuntu) is another body that has helped, over the years, to clarify the ownership of software. In order for a software package to be included in the Debian distribution, it is required to create a document that lists (among other facts) its copyright owners, and any source–if applicable–from where the package might have been derived [13]. This task is usually done manually by Debian Maintainers. For example, Figure III shows a message sent by a Debian Maintainer to the Hugin[2] developers mailing list. This message is important for three reasons: first, it shows that Debian is concerned that every file should have copyright authors; second, that proper attribution should be given to the rightful owners of a file; and third, that when a file is copied for

[2]hugin.sourceforge.net

another source, new owners can be added to it to reflect any changes made from its original.

```
I am in the process of trying to prepare 0.8.0 for Debian
GNU/Linux I have started going over the copyright/license
headers. In src/celeste many files are missing copyright
information. Most of these are files imported with minimal
changes from Gabor API http://www.kung-foo.tv/gaborapi.php
or libsvm http://www.csie.ntu.edu.tw/˜cjlin/libsvm/.

The attached patch adds copyright and license statements
to these files.[1]

Please apply and update the headers (adding copyright
holders) if you make substantial changes.

thanks, cu andreas

[1] I have doublechecked with Gabor API's upstream author
Adriaan Tijsseling that files like ContrastFilter.cpp
are Copyright (c) Adriaan Tijsseling and licensed under
GPLv2+, although the original headers just say:

     Original Author:        Yasunobu Honma
     Modifications by:       Adriaan Tijsseling (AGT)
```

Figure 1. Example of a message sent by a Debian maintainer asking developers of the Hugin project to properly state the copyright of various files.

Thus the question *"who are the copyright owners of a FOSS?"* can be answered by extracting the copyright owners of each of its files. While this appears to be a simple task, it is made difficult for the following reasons (exemplified in Figure 2[3]):

1) Is an author or a contributor a copyright owner (or co-owner) of a file? Some license statements explicitly state the "original" owner and a list of further contributors, but do not indicate if the contributors have any claim on the copyright of the system. Some license statements explicitly indicate that the contributors are the copyright owners. In other cases, the copyright owners and the authors are different persons.
2) There is no standard way to indicate the name of copyright owners or contributors in open source. We have found many variants, such as: full name (*e.g.*, *Christian González*), first name only (*e.g.*, *Christian*), a username (*e.g.*, *chrisg*), an email address (*e.g.*, *chris@corp.net*).
3) Before parsing a file, it is not known how many copyright owners or contributors are present in it, and how each one of them is separated from the rest. We have found several variants:
   - Multiple owners or contributors are listed in the same paragraph. In this case, there is no uniform character separating them. Comma, hyphen, semicolon, and periods are commonly used to separate owners. Unfortunately, what is a separator in one file might be used with a different purpose in another.
   - Each owner or contributor is listed in one or more lines.
4) Each owner or contributor is listed along with his/her contributions to the file, in something that looks more like a Changelog than a list of contributors.
5) The same author might be referred to in different ways. This could be the result of aliases (*e.g.*, *Chris*, *Christian*), misspellings (*e.g.*, *Christina*, *Cristian*, *Chrs*), one of many of his/her email addresses (*e.g.*, *Chris.Smight@gmail.com*, *chris@corp.net*). Special and accented characters are sometimes replaced by non-accented (*e.g.*, *Gonzalez* instead of *González*). There exists a risk that the names of two persons could be confused as a misspelling of the other.
6) The copyright statement might not be where the owners are actually listed. For example, in Eclipse JDT, the copyright year is usually followed by the string: "*IBM Corporation and others.*". *Others* are expected to be listed at the end of the licensing statement, under the *Contributors:* section. Mozilla does something similar: *"Portions created by the Initial Developer are Copyright (C) <year> the Initial Developer. All Rights Reserved."* Another section of the license contains the name of the *Initial Developer*, and a list of contributors to the file. Apache HTTPd takes an ever more radical approach, by stating that some owners are listed in a file named `NOTICES`.

As described above, license statements frequently distinguish between the original copyright owner of an artifact (in this case a file) and its subsequent contributors. According to copyright law, all of them are co-owners of the file, but each owns his/her own contribution as long as such contribution is considered "copyrightable" (like many definitions in copyright law, it does not qualify what copyrightable means in terms of an objective metric, but rather subjectively). Thus, it is necessary to detect and extract both original authors, and contributors of a file. To avoid confusion, and to clarify our nomenclature, we will use the following terms in the rest of this paper:

- *Contributor:* a person or organization whose name appears either as initial author or contributor in the license statement.
- *Explicit Copyright Owner:* a person or organization whose name appears explicitly stated as a copyright owner by immediately following the "Copyright <year>" string in a license statement.
- *Committer:* A person who commits code to the version control system. One person might have more than one committer id in the version control system. This is common in Mozilla, where the committer id reflects the *current* email address of the developer; when a developer changes email address, a new id is created. For example, *aaronleventhal%moonset.net* and *aaronl%netscape.com* correspond to the same person.

[3]To provide a comprehensive set of examples, we also show cases for projects that, due to space limitations, are not part of the empirical study described in this paper.

Apache HTTPd

```
/* Licensed to the Apache Software Foundation (ASF)
under one or more
 * contributor license agreements.  See the NOTICE
file distributed with
 * this work for additional information regarding
copyright ownership.
[...]
 * Code originally by Rob McCool; much redone by
Robert S. Thau
 * and the Apache Software Foundation.
```

Squid

```
 *  Copyright 1997 by Carnegie Mellon University
 *  All Rights Reserved
 *  Permission to use, copy, modify, and distribute
 *  this software and its documentation for any purpose
 *  and without fee is hereby granted,
[...]
 *
 * CMU DISCLAIMS ALL WARRANTIES WITH REGARD
 * TO THIS SOFTWARE, INCLUDING
[...]
 * Author: Ryan Troll <ryan+@andrew.cmu.edu>
```

Mozilla

```
 * The Initial Developer of the Original Code is
Netscape Communications Corporation.
 * Portions created by the Initial Developer
are Copyright (C) 2002
 * the Initial Developer. All Rights Reserved.
 *
 * Contributor(s):
 *  Simon Fraser <smfr@smfr.org>
 *  Stuart Morgan <stuart.morgan@alumni.case.edu>
```

Eclipse JDT

```
* Copyright (c) 2000, 2009 IBM Corporation and others.
* All rights reserved. This program and the
[...]
* Contributors:
*   IBM Corporation - initial API and implementation
*   Tom Eicher <eclipse@tom.eicher.name> - [formatting]
Format Element' in JavaDoc does also format method body -
https://bugs.eclipse.org/bugs/show_bug.cgi?id=238746
*     Tom Eicher (Avaloq Evolution AG) - block
selection mode
```

Figure 2. Examples of copyright attributions as they exist in a source file in different projects. Lines that do not start with * are continuation of the previous one.

The copyright owners of a file would be the explicit copyright owners, plus the contributors to the file. Clearly, there could be contributors that change a substantial portion of the file just because of a bug fixing or of a refactoring. The question whether these contributors should also be copyright owners is still open and questionable. The *copyright owners of a system* would be the union of the copyright owners of each of its files. There are other copyright owners, such as those that create artwork and manuals; in this paper we will focus our attention to source code developers only.

IV. A METHOD TO DETERMINE WHEN CONTRIBUTORS ARE ADDED TO A FILE

We have developed a method to a) extract and uniquely identify contributors, b) determine when contributors are added to or removed from a file, and c) whenever possible, match contributors to version control committer ids. It consists in five steps that we detail below. Step 1 aims at extracting licensing statements from source code files. Step 2 aims at identifying contributor names within licensing statements. Step 3 identifies different ways in which the same contributor is listed. Step 4 maps contributor names to committer ids from the version control logs. Finally, Step 5 compares the set of contributors for each file revision with its previous one, identifying the contributors that have been added or removed in that revision.

Step 1: Extracting licensing statements from source code files. Usually a licensing statement is located in the first blocks of comments of a file (where a block is a sequence of consecutive comments). We created our own comment extractor based on a comment-removal tool adapted to export comments instead of removing them[4]. We found that, usually, the licensing statement is the first block, but in rare occasions is located in the second block of comments, or span both first and second (*e.g.*, because it is interleaved with preprocessor directives). For this reason, we always extracted the first two comment blocks.

Step 2: Extracting contributors. This is the core of our data extraction process. Its goal is to mine contributor names into the licensing statements extracted in the previous step. It, in turn, consists of the following tasks:

1) Extract the contributors section from the licensing statement. The main challenge faced in this step is that each project uses a different format to credit its contributors, and therefore, should be adapted accordingly. Mozilla, as shown in Figure 2, adds at the end of its licensing statement a section called *"Contributor(s):"*, which ends with the string *"Alternatively, the contents [...]"*. The steps followed to extract the contributors at each revision are: as the area delimited from *"Contributor(s):"* until *"Alternatively, the contents [...]"* is found or until the end of the licensing statement. ArgoUML and Squid often mention the contributor names after the keyword *"@author"* and *"Author:"* respectively. Last, but not least, all four projects contain copyright statements where copyright years and contributors are also mentioned. Examples are *"Copyright (C) Tim Potter 2000"* (Samba), or *"Copyright (C) 1999,2002 Henrik Nordstrom <hno@squid-cache.org>"* (Squid). Regular expressions to match copyright statements were defined after manually inspecting a sample of about 500 files

[4]Our comment extractor can be downloaded from turingmachine.org/~dmg/comments-1.0.tar.bz2

from the four projects. It is also important to mention that, in some cases, a file contains both a copyright belonging to an institution (*e.g.*, many files of Squid are copyrighted by the Carnegie Mellon University) and an author listed using one of the criteria above mentioned. In that case we considered both the institution and the author as contributors.

2) Break each contributors section into sentences, removing unnecessary whitespace and commenting characters from the input, saving the result into a file. We will call this the set of contributors for the file revision.
3) Remove duplicates from the previous step.
4) Filter out unnecessary information (such as the explanation of the contribution) and to make sure that only one contributor was listed per each line.

For example, the following contributors section:

```
* Keith Visco, kvisco@ziplink.net
 * -- original author.
 *
 * Nathan Pride, npride@wavo.com
 * -- fixed document base when stylesheet is specified,
 *    it was defaulting to the XML document.
 *
 *Olivier Gerardin, ogerardin@vo.lu
 *  -- redirect non-data output (banner, errors) to stderr
 *  -- read XML from stdin when -i is omitted
 *  -- accept '-' to specify stdin/stdout on command line
```

was manually transformed to

```
Keith Visco <kvisco@ziplink.net>
Nathan Pride <npride@wavo.com>
Olivier Gerardin <ogerardin@vo.lu>
```

Step 3: Identify variants in names, spelling mistakes, and unify contributor names. It often happens that, in different files, the same contributor is listed in different ways, for example, in Mozilla *"Christopher A. Aillon"* or *"Christopher Aillon"*; *"IBM Corp"*, *"IBM Corporation"*, or *"International Business Machines"*; in other occasions there can be spelling mistakes (*"Blizzzard"* instead of *"Blizzard"*). To deal with these inconsistencies, we build a thesaurus of contributor names, where each contributor has a unique identifier (possibly her full name), and other names are treated as aliases. Where available, other information is associated to the contributor, *i.e.*, the email address and the company who employed him/her. This process is performed manually also.

Step 4: Match contributors to Concurrent Versions System (CVS)/SubVersioN (SVN) ids. We use a method that is an extension of Bird's algorithm to match email addresses to committer ids [14]. For example, Mozilla uses as committer ids of its CVS repository the email addresses of the individuals (replacing `'@'` with `'%'`). This means that one person might have several CVS ids (for example, Aaron Leventahl has the following ids: *"aaronl%chorus.net"*, *"aaronleventhal%moonset.net"*, and *"aaronl%netscape.com"*). We assume that, if the string before the % is the same, then the id belongs to the same person.

For the other systems we analyze the mapping was relatively simpler, primarily because they had significantly fewer CVS/SVN committers and contributors. In contrast with Mozilla, in the other systems each contributor used only one CVS/SVN id. Basically, two situations occurred:

- the contributor name matches the CVS/SVN id, *e.g.*, in ArgoUML *"aslo"* or *"euluis"* are both contributor names and SVN ids.
- the CVS/SVN id is a shortcut of the contributor name (*e.g.*, initials), and there is no ambiguity with other contributors, e.g. in ArgoUML *"Jeremy Jones"* $\rightarrow$ *"jjones"*, or in Samba *"Andrew Barteltt"* $\rightarrow$ *"abartlet"*.

Step 5: Identifying when the set of contributors of a file changes. The final step compares the set of contributors of each file revision with the set of contributors of the previous revision. The result is, for each file revision, the set of contributors added and removed.

V. Empirical study

The *goal* of this study is to analyze the set of source code files contributors, as stated in licensing statements, with the *purpose* of investigating how this set changes, and whether such changes are related to two reasons: the amount of code the contributor commits (when the owner is also a committer), and the number of times that this person has committed to such file. The *quality focus* is related to the ownership of source code files, and to how such ownership changes. The *perspective* is of researchers who want to understand in what context do the ownership of a source code file changes. The ownership of a file will affect whether this file can change its license in the future. The *context* consists of the CVS or SVN repositories of four FOSS systems: ArgoUML, Mozilla, Samba, and Squid. The four systems have different sizes, were developed with different programming languages (Java, C++, and C), and belong to different domains: ArgoUML is a UML modeler; Mozilla is a suite comprising a Web browser, an email client, and other Internet utilities; Samba is a file and printer service interoperating between Unix and Windows operating systems; and Squid is a Web proxy server. The version control used in the projects we investigated is CVS, except for ArgoUML, which uses SVN[5]. Table I reports the main characteristics of the four systems.

A. Research questions

The empirical study aims at addressing the following research questions:

RQ1: *Who are the contributors of source code files?* This research question aims at mining contributors of each source code file, at analyzing in how many files each contributor

[5]For Samba, we chose the time interval when it was still under CVS. It later migrated to SVN and finally Git.

Table I
MAIN CHARACTERISTICS OF THE FOUR SYSTEMS.

Characteristic	ArgoUML	Mozilla	Samba	Squid
Language	Java	C/C++	C	C
Release range	0.10–0.20	M3–1.7.13	1.9–3.0	1.0–3.0
#of source files range	777–1,421	4,845-12,436	299–860	21–876
KLOC range	129–280	1,827–4,104	156–332	13–180
CVS/SVN start date	2000-09-14	1998-03-28	1996-05-04	1996-02-22
CVS/SVN end date	2005-12-30	2008-01-11	2004-04-03	2008-03-03
Analyzed file revisions	32,582	468,747	29,018	6,359

is listed. It also investigates whether contributors are also committers in the versioning system.

RQ2: *How frequently do contributors change?* This research questions investigates the change of contributors in source code files, specifically analyzing how frequently contributors are added, and if sometimes contributors are removed from a file licensing statement.

RQ3: *When do contributors change?* This research question investigates whether people appearing as file contributors have changed, on average, a number of source code lines higher than other committers, and if there a relationship between large changes and the addition of contributor names to source code files.

B. Analysis method

To answer **RQ1**, first we analyze the percentage of files that, during their lifetime, had at least one contributor mentioned in the licensing statement, vs. the percentage of files for which no contributor was ever mentioned. This to understand to what extent source code files of a project contain contributor information. Then, we investigate the mapping existing between contributor names and CVS/SVN committer ids, *i.e.*, how many contributors are mapped to committers, and *vice versa*. Finally, we analyze the distribution of contributors across source code files, by highlighting contributors who appear on a larger number of files than others.

To answer **RQ2**, we compute, for each file, the percentage of commits that contain an addition or a removal of contributors, and analyze the distribution of such percentages. Since in this context we are mainly interested in changes in the list of contributors, rather than on the presence of the contributors themselves, we do not account for cases where the addition occurs in the first file revision (*i.e.*, 1.1), since for that cases we consider that the contributor was always mentioned in the file, until a further change occurred.

To answer **RQ3**, we look at various factors that could be related to the addition of contributors into licensing statements. In particular, we investigate:

- whether contributors are introduced in the context of substantial changes occurring to source code files, *i.e.*, changes that, in terms of number of lines added/removed, are larger than other changes. The rationale here is to verify if a large change results in adding contributors to the file. The comparison is performed using (non-parametric) Mann-Whitney test, and the magnitude of the difference is estimated using the Cohen d effect size [15], which is defined as the difference of means divided by the pooled standard deviation, and is considered small for $0.2 \leq d < 0.5$, medium for $0.5 \leq d < 0.8$ and large for $d \geq 0.8$.
- whether the contributors of a file performed, over the file lifetime, an amount of changes (in terms of number of commits or lines added/removed) larger than other committers. The rationale is that we would expect that those who contributed more to a file would be listed as contributors. Here the comparison is performed using a Wilcoxon test (which does a pairwise difference on each file of the changes made by contributors and by others), and the magnitude of the difference computed using the Cohen d effect size for dependent samples, defined as average difference across samples (files in our case), divided by the pooled standard deviation.

Table II
NUMBER OF FILES WITH AND WITHOUT CONTRIBUTORS.

System	TOTAL	With contrib	Without contrib
ArgoUML	4795	1390 (29%)	3405 (69%)
Mozilla	16763	3224 (19%)	13539 (81%)
Samba	1389	618 (44%)	771 (56%)
Squid	774	248 (32%)	627 (68%)

VI. RESULTS

This section reports and discusses results aimed at addressing research questions formulated in Section V-A. To favor replication, raw data used for the analyses are available for downloading[6].

A. RQ1: Who are the contributors of source code files?

To get a first idea of how contributors appear in source code files, we identify the number (and percentage) of source code files for which at least one contribution was mentioned in at least one revision of the source code file. Results are reported in Table II. As it can be noticed from the table, in the four analyzed projects the presence of contributor names

[6]http://www.rcost.unisannio.it/mdipenta/contrib-data.tgz

Table III
NUMBER OF CONTRIBUTORS (CT), COMMITTERS (CM) AND THEIR INTERSECTION AND DIFFERENCE.

System	Ct	Cm	$Ct \cap Cm$	$Ct - Cm$	$Cm - Ct$
ArgoUML	60	40	25	35	15
Mozilla	732	597	274	458	323
Samba	55	35	19	26	16
Squid	42	10	7	35	3

in source code files is not negligible, although the majority of files do not contain contributor information. In particular, the percentage is particularly low for Mozilla: (19%), despite the well-defined format Mozilla has adopted for mentioning contributors. On the other hand, almost every Mozilla file cites the Mozilla Foundation or Netscape Corporation as its initial author. Since we are interested to analyze how contributors change, we do not include the original author of Mozilla files in our study if it is the Mozilla Foundation or Netscape Corporation.

Table III reports, for each system, (i) the overall number of contributors (Ct), (ii) the overall number of committers to its version control repository (Cm), (iii) the number of cases in which a contributor was also a committer ($Ct \cap Cm$), (iv) the number of contributors that are not committers ($Ct - Cm$), and (v) the number of committers that are never mentioned as contributors ($Cm - Ct$). The table shows that:

- Mozilla has a very large number of contributors and committers. As discussed in Section IV, for Mozilla a contributor can appear in the CVS with different identifiers. All the 597 committers of the files we analyzed result mapped to 274 contributors (*i.e.*, some contributors correspond to more than one committer).
- Squid has 4 times more contributors than committers, and 70% of the committers are listed as contributors of at least one file. For Samba and ArgoUML, the number of contributors is about 50% more than the number of committers, and more than half of these committers are listed as contributors to at least one file.

Table IV shows the number of files on which the most frequent contributors for the four systems appear. We indicate in boldface contributors that are also committers. As it can be seen, with few exceptions, the most common contributors are also committers. In addition, there is a large number of files owned by organizations, in particular IBM Corp., Netscape Corp., and Carnegie Mellon University; this is common for code owned by companies, where often the company name appears in place of the author. It also reflects that some contributors act on behalf of their employer.

B. RQ2: How frequently do contributors change?

In this research question we analyze to what extent contributors change across file revisions. Figure 3 shows, for files belonging to the four analyzed systems, boxplots of frequencies of contributors additions. The analysis only includes the set of files for which at least one contributor was found. The frequency is significantly different among systems (p-value < 0.001), with median values of 4% for Mozilla, 5% for Samba and Squid, and 8% for ArgoUML. Mozilla is the only system for which we detected contributor removals. It is also interesting to note that in Mozilla we detected commits where contributors were removed. This happened with a median frequency of 3%. Usually these removals were "rollbacks" to previous file revisions, and the these removed contributors would be latter added. We presume that rollbacks were the result of defects in such commits, and once the code was improved their contribution was added again to the project, and their name to the list of contributors.

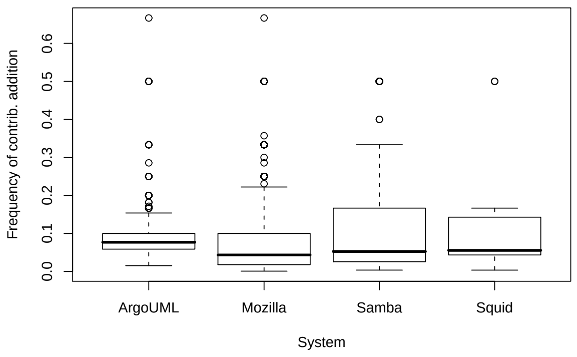

Figure 3. Frequency of contributors addition.

C. RQ3: When do contributors change?

RQ3 aims at investigating in what context new contributors are added to source code files. In particular, we analyze whether contributors are added in the context of larger changes to a source code file. To this aim, we compare the number of lines added/deleted—as extracted from the versioning system—for commits when contributors were added to files, with the same number for other commits (such a number is normalized dividing it by the file size at the time of the commit). Results are shown in Table V. It can be noticed that contributors are added in the context of commits that, on average, are significantly larger—and with a medium/high effect size—than other commits, except for ArgoUML, for which there is a significant difference in the opposite direction, although with a negligible effect size.

Then, we investigate whether contributors are those committers who perform a larger amount of changes on source code files. In other words, it could be expected that, after a certain amount of changes, a committer deserves to add his/her name to the source code file. To this aim, we compared the number of lines added/removed to source code files by committers that have been contributors at

Table IV
MAJOR CONTRIBUTORS IN TERMS OF NUMBER OF FILES THEY APPEAR ON (BOLDFACE HIGHLIGHTS WHO IS ALSO A COMMITTER).

ArgoUML		Mozilla		Samba		Squid	
Jaap Branderhorst (kataka)	**443**	**Pierre Phaneuf**	**456**	**Andrew Tridgell**	**350**	**Robert Collins**	**128**
Bob Tarling	**154**	IBM Corporation	139	**Tim Potter**	**152**	Carnegie Mellon University	26
Thierry Lach	**150**	**Darin Fisher**	**130**	**Luke Kenneth Casson Leighton**	**128**	**Duane Wessels**	**16**
Jason Robbins	**101**	**Stuart Parmenter**	**93**	**Jeremy Allison**	**119**	Harvest Derived	14
MVW	**85**	**Brian Ryner**	**90**	**Andrew Bartlett**	**79**	**Guido Serassio**	**11**
mkl	**75**	**Keith Visco**	**86**	Paul Ashton	57	Ryan Troll	11
alexb	**53**	Netscape Communications Corporation	85	**Jerald Carter**	**56**	Francesco Chemolli	6
Curt Arnold	**47**	**David Hyatt**	**78**	**Jim McDonough**	**51**	Flavio Pescuma	5
Linus Tolke	**44**	**Seth Spitzer**	**74**	Simo Sorce	47	**Henrik Nordstrom**	**5**

Table V
AMOUNT OF CHANGES WHEN CONTRIBUTORS ARE ADDED VS. OTHER COMMITS (VALUES ARE NORMALIZED WITH RESPECT TO THE FILE LENGTH).

System	Contr. added		No contr. added		Mann-Whitney	Effect
	Mean	σ	Mean	σ	p-value	size
ArgoUML	0.11	0.19	0.15	4.63	**0.01**	-0.01
Mozilla	0.22	0.53	0.05	0.20	$<$ **0.01**	0.75
Samba	0.73	1.17	0.07	0.59	$<$ **0.01**	1.09
Squid	1.55	3.98	0.10	0.72	$<$ **0.01**	1.91

least for a period of the file lifetime, with the number of lines added/removed by other committers (that were not contributors to that file). Descriptive statistics, Wilcoxon test p-values[7], and the Cohen d effect size, are reported in Table VI. As the table shows, the results is significant in the opposite direction, *i.e.*, the amount of changes made by non-contributors is significantly higher than the amount of changes made by contributors. This means that important contributors are not mentioned in source code files. Further investigation is needed to understand why this does not happen.

D. Threats to Validity

This section discusses the main threats to the validity of our study. *Construct validity* threats concern the relation between the treatment and the outcome. They can be due to our measurements, *i.e.*, in particular to the performance of the heuristics we used to extract contributors from source code files. Although our results are not affected by problems of accuracy (as the list of extracted contributors was manually scrutinized), the heuristic we used could have missed some contributors. We limited this problem by carefully analyzing several file headers of all the software systems, to determine the heuristics and regular expressions for extracting contributor names. Another threat is related to the fact that, like Gîrba *et al.* [8], we measured the contribution of a committer in terms of lines added/removed. We are aware that, in many cases, simple code style improvements or formatting can be seen as addition/removal of source code lines. In future work we plan to use more sophisticated differencing tools (e.g., [16]) to better identify the changes being performed.

Threats to *internal validity* do not affect this study, being this an exploratory study. For the same reason, threats to *conclusion validity* are also not important, although we used statistical tests where appropriate and made sure that the conditions for their applicability held.

Threats to *external validity* are related to the generalizability of our findings. Our study includes four open source systems, developed in different programming languages, belonging to different domains, and experiencing different kinds of evolution, *e.g.*, systems developed from scratch (ArgoUML, Samba, and Squid), and another that originated in industry (Mozilla). Yet, it is necessary to replicate this study on other systems.

Regarding *reliability validity*, *i.e.*, the possibility of replicating this study, we have detailed the data extraction process, and the source code and changes for the four systems are available from their CVS/SVN repositories. Furthermore, we make available the extracted data to ease the replication of our analyses.

[7]As in this case the comparison is pairwise on each file, we do not need to normalize the change dividing it by the file size.

VII. RELATED WORK

To the best of our knowledge, only few recent works investigate the activity of developers in software projects and, above all, how they are mentioned in various software repositories.

The related work closest to ours is that of Robles and González-Barahona [17], who outlined a process to get identities from different sources (commits, mailing lists, bug reports, source code), and to map them. With respect to their work, we propose precise heuristics to (i) identify contributors within source code files and to map them to committers, and (ii) we report an empirical study across four systems investigating various questions about the presence of contributor names in source code files. German [7] studied the activity of PostgreSQL committers. He discovered that a large number of commits where authored by people without commit access, and that one particular person was responsible for committing such contributions. Hindle *et al.* [9] discovered that many of the largest commits correspond to changes to the licenses or copyright owners of files. This

Table VI
NUMBER OF SOURCE CODE LINES ADDED/REMOVED BY CONTRIBUTORS AND BY OTHER COMMITTERS.

System	Contributors			Other committers			Mann-Whitney	Effect
	Mean	Median	σ	Mean	Median	σ	p-value	size
ArgoUML	1.18	0.00	19.74	178.07	45.00	574.36	$<$ **0.01**	0.31
Mozilla	105.81	0.00	1400.16	795.92	74.00	8210.58	$<$ **0.01**	0.08
Samba	262.07	0.00	921.28	945.19	116.00	2374.06	$<$ **0.01**	0.33
Squid	12.47	0.00	88.32	283.07	12.00	2190.94	$<$ **0.01**	0.12

confirms what we quantitatively found in RQ3, *i.e.*, new contributors were added to files in the context of large changes.

Other works were specifically related to social network analysis of developers' activities. Yu and Ramaswamy [18] used clustering techniques on interactions among developers to identify developers roles, *i.e.*, classifying developers into *core* and *associate* developers, according to the strength of their collaborations. Differently from our work, the aforementioned works focus on committers rather than on contributors appearing into source code files. Bird *et al.* [19], [20] developed a method to unify email accounts coming from mailing lists, where often the same developer tend to used multiple accounts. Then, they performed social network analyses from mailing lists coming from several open source projects. They found that the network of collaboration was in most cases modular when the collaboration was related to specific software artifacts, and that the division of the project in sub-communities reflected the technical content of the discussion. Again, the previous studies focused on committers or mailing list recipients; we believe that further developer/contributor collaborations can be inferred by exploiting contributors extracted from source code files.

Ownership was computed by Gîrba *et al.* [8] as the percentage of source code lines modified by a specific author. They also pointed out the need for properly visualize the ownership of a file, and defined a view named *Ownership Map* where changes to files were represented with circles having a radius proportional to the change size and a color representing the owner. We share with them the idea of ownership measured in terms of amount of changed lines, while we found that this do not imply (RQ3) having the contributor mentioned in the file. Instead, results showed that contributors were added in large changes.

Other works related to legal issues in open source projects focus on licenses. Gobeille [21] developed FoSSology, a tool to automatically classify open source licenses using a pattern matching algorithm called the Binary Symbolic Alignment Matrix (bSAM). FoSSology is capable of detecting 78 different license variants, classified in a hierarchy of licenses (for example, there exist several kinds of Corporate licenses belonging to different companies, as well as different versions of the GPL). Licenses impose constraints and thus can be defined as logical formula constraining what can and cannot be done with a system. Software licensing patterns have been recently studied by German *et al.* [4] using such a formalization of licenses. They introduced several legal patterns, along with examples of occurrences of these patterns. Finally German *et al.* [12] presented a study of the influence of software licenses on code migration between the FreeBSD, Linux, and OpenBSD kernels.

VIII. CONCLUSIONS AND WORK-IN-PROGRESS

Source code file contributor names, together with licenses, represent important software assets, as they influence the way source code can be used. As the signature on a picture, they indicate the intellectual property of the code. Also, they can both represent people who are directly involved in source code development—and thus they are mentioned as authors are mentioned on a paper—or can represent people who contributed indirectly and deserve credit, as it happens for people acknowledged at the end of a paper. It is not only interesting to monitor contributors to see how people contribute to a software project, but also to make sure that when the source code is cloned from a system to another [12] ownership and credits are preserved. Also, contributor names mined from source code files are an alternative to committer ids to determine competent source code file developers for the purpose of building triaging systems [22], and for analyzing socio-technical aspects of software development [20].

This paper proposed a method, based on both (semi) automatic analysis of source code comments and on a manual mapping to committer identifiers, to extract contributor names from source code files and map them to committers. Then, it reports an exploratory study aimed at analyzing the presence and the evolution of contributor names in the source code of four open source projects, namely ArgoUML, Mozilla, Samba, and Squid. Results indicate that:

- Matching contributors to committer ids is not trivial. For example, in Mozilla many individuals have more than one committer id; in some cases, the committer id had little resemblance to the name of the committer (such as Samba contributor *Jaap Branderhorst*, who has CVS id *kataka*, which has no resemblance to his email address id jaap.branderhorst).
- On average, less than one third of the source code files mention, in at least one revision, their contributors; in some cases, organizations tend to be mentioned instead of people (*e.g.*, corporations such as IBM, Netscape, or

universities such as Carnegie Mellon University),
- The analyzed software projects have many more contributors than committers, although almost all committers are also mentioned as contributors (while not all contributors are committers).
- Contributors tend to be changed in less than 10 percent of the file revisions.
- While contributors that appear on a large number of files are also committers, the amount of changes performed by contributors is not higher than that of other committers. This indicates that some important committers are not mentioned as contributors in source code files.
- The addition of a new contributor occurs in the context of changes significantly larger than others, *i.e.*, a large change tend to be recognized by the name mentioned in the file.

Work-in-progress is complementing this activity of analyzing contributors, with analyses related to introduction and change of licenses in source code files, to see to what extent a source code file, during its lifetime, (i) changes the list of contributors, (ii) gets the copyright years updated, and (iii) moves from a license (*e.g.*, BSD) to another (*e.g.*, GPL), or gets its license updated.

References

[1] T. Golder and A. Mayer, "Whose IP is it anyway?" *Journal of Intellectual Property Law & Practice*, vol. 4, no. 3, pp. 165–175, 2009.

[2] A. M. S. Laurent, *"Understanding Open Source and Free Software Licensing"*. O'Reilly, 2004.

[3] L. Rosen, *Open Source Licensing: Software Freedom and Intellectual Property Law*. Prentice Hall, 2004.

[4] D. M. Germán and A. E. Hassan, "License integration patterns: Addressing license mismatches in component-based development," in *31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Proceedings*. IEEE, 2009, pp. 188–198.

[5] Free Software Foundation, "Frequently Asked Questions about the GNU Licenses," http://www.fsf.org/licensing/licenses/gpl-faq.html, accessed Feb. 2009.

[6] M. Vlimki, "Dual Licensing in Open Source Software Industry," *Systemes d Information et Management*, vol. 8, no. 1, pp. 63–75, 2004.

[7] D. M. Germán, "A study of the contributors of PostgreSQL," in *Proceedings of the 2006 International Workshop on Mining Software Repositories, MSR 2006, Shanghai, China, May 22-23, 2006*. ACM, 2006, pp. 163–164.

[8] T. Gîrba, A. Kuhn, M. Seeberger, and S. Ducasse, "How developers drive software evolution," in *8th International Workshop on Principles of Software Evolution (IWPSE 2005), 5-7 September 2005, Lisbon, Portugal*. IEEE Computer Society, 2005, pp. 113–122.

[9] A. Hindle, D. M. German, and R. Holt, "What do large commits tell us? a taxonomical study of large commits," in *MSR '08: Proceedings of the 2008 international working conference on Mining software repositories*, May 2008, pp. 99–108.

[10] World International Property Organization, "CRNR/DC/94 WIPO Copyright Title," Dec 1996.

[11] MySQL AB, "MySQL Contributor License Agreement v0.3," http://forge.mysql.com/contribute/cla.php, accessed Sept. 2008.

[12] D. M. German, M. Di Penta, Y.-G. Guéhéneuc, and G. Antoniol, "Code siblings: Technical and legal implications," in *Proc. of the 2009 Working Conference on Mining Software Repositories, MSR 2009*, 2009.

[13] I. Jackson and C. Schwarz, "Debian Policy Manual," 1998, http://www.debian.org/doc/debian-policy/.

[14] C. Bird, A. Gourley, P. T. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks in Postgres," in *Proceedings of the 2006 International Workshop on Mining Software Repositories, MSR 2006, Shanghai, China, May 22-23, 2006*. ACM, 2006, pp. 185–186.

[15] J. Cohen, *Statistical power analysis for the behavioral sciences (2nd ed.)*. Hillsdale, NJ: Lawrence Earlbaum Associates, 1988.

[16] G. Canfora, L. Cerulo, and M. Di Penta, "Tracking your changes: a language-independent approach," *IEEE Software*, vol. 27, no. 1, pp. 50–57, 2009.

[17] G. Robles and J. M. González-Barahona, "Developer identification methods for integrated data from various sources," in *Proceedings of the 2005 International Workshop on Mining Software Repositories, MSR 2005, Saint Louis, Missouri, USA, May 17, 2005*. ACM, 2005.

[18] L. Yu and S. Ramaswamy, "Mining CVS repositories to understand open-source project developer roles," in *Fourth International Workshop on Mining Software Repositories, MSR 2007, Minneapolis, MN, USA, May 19-20, 2007, Proceedings*. IEEE Computer Society, 2007, p. 8.

[19] C. Bird, A. Gourley, P. T. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *Proceedings of the 2006 International Workshop on Mining Software Repositories, MSR 2006, Shanghai, China, May 22-23, 2006*. ACM, 2006, pp. 137–143.

[20] C. Bird, D. S. Pattison, R. M. D'Souza, V. Filkov, and P. T. Devanbu, "Latent social structure in open source projects," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2008, Atlanta, Georgia, USA, November 9-14, 2008*, pp. 24–35.

[21] R. Gobeille, "The FOSSology project," in *In Proc. Fith International Workshop on Mining Software Repositories*, 2008, pp. 47–50.

[22] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in *28th International Conference on Software Engineering (ICSE 2006), Shanghai, China, May 20-28, 2006*. ACM, 2006, pp. 361–370.