A sentence-matching method for automatic license identification of source code files

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

Yuki Manabe
Osaka University, Japan
y-manabe@ist.osaka-u.ac.jp

Katsuro Inoue
Osaka University, Japan
inoue@ist.osaka-u.ac.jp

ABSTRACT

The reuse of free and open source software (FOSS) components is becoming more prevalent. One of the major challenges in finding the right component is finding one that has a license that is adequate for its intended use. The license of a FOSS component is determined by the licenses of its source code files. In this paper, we describe the challenges of identifying the license under which source code is made available, and propose a sentence-based matching algorithm to automatically do it. We demonstrate the feasibility of our approach by implementing a tool named `Ninka`. We performed an evaluation that shows that `Ninka` outperforms other methods of license identification in precision and speed. We also performed an empirical study on 0.8 million source code files of Debian that highlight interesting facts about the manner in which licenses are used by FOSS.

1. INTRODUCTION

Free and Open Source Software (FOSS) has become an important source of reusable code [19]. To be able to reuse a FOSS component an application (proprietary or FOSS) should satisfy all the requirements and conditions that the license of the component imposes [17]. Sometimes a component is available under various licenses, giving the integrator the ability to choose the license that best fits her purpose (the license of the component is compatible with the intended use). For example, the MySQL database can be used under the conditions of the General Public License (GPL) version 2, or the integrator can buy a commercial license for her specific needs [9]. Most frequently, however, a component can only be used under one license. For example, in August of this year, Nokia tried to convince the developers of PyQT to change its license from GPL version 2 to the Lesser GPL version v2.1, but did not succeed. As a result, Nokia has started a project to create a replacement, called PySide[15]. Nokia explains its rational: *"PySide has been published as a response to the lack of suitably licensed Qt bindings for Python. PySide is licensed under the LGPL version 2.1 license, allowing both Free/Open source software and proprietary software development."*

In an empirical study, Li et al. reported that 37% of companies that used OSS components modify their source code [14]. They also emphasize that any organization wanting to reuse FOSS components (either with or without modification) should consider the legal implications of such changes.

The legal issues of reusing FOSS components affect not only companies, but other FOSS components and applications. As reported in [9], FOSS applications are also concerned about licensing issues, primarily if the license of the FOSS component is compatible with the license of the application that uses it. If they are not, then the component cannot be used.

One of the major challenges of intellectual property clearance is to identify the license under which a FOSS component, and each of its files, is made available. This is due to several factors: 1) there is a vast number of open source licenses, some approved by the Open Source Initiative (currently 65), and many more that are not—Table 1 shows some frequent FOSS licenses and their abbreviations, as used in this paper; 2) a FOSS product might be made available under several licenses, 3) different versions of a FOSS component might be available under different licenses; and 4) the overall license of a product might be different than the license of each of its files.

When files under various licenses are combined to create a FOSS component, there is a risk that one of these licenses might be incompatible with the overall license of the component, hence any integrator using that component might be liable for infringement [17, 18].

Another problem is that several well-known licenses exist in various versions. For example, the GPL exists in versions 1, 2 and 3, and each of these versions is incompatible with the others (files under one version cannot be combined with files under another version, as described in [7]); similarly code under the *BSD 4* clauses code cannot be included in software released under the *BSD 3* or *BSD 2* clauses. Frequently FOSS software portals, such as SourceForge, only use name of the license and omit the version in the metadata of the applications they host. Furthermore, they make no attempt to verify that this license listed is indeed the license of the component. For example, the FOSS portal Freshmeat (`freshmeat.net`) lists Eclipse as licensed under the CPL, when in fact, it uses the EPLv1.

The contributions of this paper are: 1) We describe and categorize the challenges of license identification. 2) We propose a method for license identification based on the analysis

Abbrev.	Name
Apache	Apache Public License
BSD4	Original BSD, also known as BSD with 4 clauses
BSD3	BSD4 minus advertisement clause
BSD2	BSD3 minus endorsement clause
boost	The Boost Software license
CPL	Common Public License
CDDL	Common Development and Distribution License
EPL	Eclipse Public License
GPL	General Public License
IBM	IBM Public License
LesserGPL	Lesser General Public License (successor of the Library GPL, also known as LGPL)
LibraryGPL	Library General Public License (also known as LGPL)
MIT/X11	Original license of X11 released by the MIT
MITold	License similar to the MIT/X11, but with different wording
MPL	Mozilla Public License
SameAsPerl	File is licensed in the same terms as Perl
SeeFile	File points to another where the its license is
SunSimpleLic	Simple license used by software developed by Sun Microsystems
ZLIBref	The zlib/libpng license

Table 1: Names of common open source licenses and their abbreviations used in this article. Many of these licenses have several versions. In that case we use the suffix v<number> to identify it. If it is followed by +, that means the user can choose this version or any newer: *"or later"*. For example, GPLv2+ means *"GPL version 2 or later"*.

of each of the sentences in the license statement of a source code file (the portion of the file that contains the terms under which the file is licensed). 3) We perform an empirical evaluation of our method and three other license identification tools showing that our method outperforms the others in precision and speed. Finally, we present an empirical study of approximately 0.8 million source code files that demonstrates the feasibility of our method and highlights some characteristics of the legal landscape of FOSS.

1.1 Related work

Several papers have used metadata from FOSS forges and application portals (such as Freshmeat and SourceForge) to identify the license of FOSS applications. In [2] Capaluppi et al. used this data to report that, in a sample of 400 applications, 77% used the GPL, 6% the LGPL and 5% the BSD (no versions were indicated). Other papers have used this data to analyze the impact of a license in the success and impact of FOSS applications [20, 4]. The authors of these papers did not analyzed its source code.

Currently two methods exist for license identification. The first is the use of regular expressions to identify the license of a file. Two tools use this method: `ohcount`, developed by Ohloh (`ohloh.net`), a portal of statistics for FOSS projects recently acquired by SourceForge; and OSLC, the Open Source License Checker (`http://oslc.sourceforge.net/`). The second method for license identification is used by FOSSology [11]. It uses an algorithm called the Binary Symbolic Alignment Matrix (bSAM), traditionally used in protein alignment. All these tools are open source.

In [21] Tuunanen et al. compared OSLC, FOSSology and their own implementation, called ASLA (based on the regular expression method). Their evaluation consisted in an empirical study that used 12 FOSS applications. They measured the recall of each of the tools, but did not verify their precision. They also reported that the major disadvantage of FOSSology is that it is slow (it took approximately 1 minute per file) compared to OSLC and ASLA (that were able to process almost 4000 files per minute in the same computer). To date, this has been the only formal evaluation of license identification tools. License identification is only a facet of ALSA. It is also a GUI environment for the analysis of the interactions between the licenses of different FOSS components in a heterogeneous system. Similar work has been proposed by Alspaugh et al. [1].

Research we have performed in the last two years has required the identification of licenses of a large collection of files (see [10, 16, 6]). We have identified the following issues with the current license identification systems:

- They are not precise enough. In many cases these systems can identify the family of a license, but not its version. In [6] and [8] we have demonstrated that license evolution is a growing issue, acerbated by lack of tools that properly identify the version of a license. As licenses evolve, the new versions are not always compatible with older versions.

- They do not report unknown licenses. It is important to know that a file has a license, even if the tool is not capable of identifying it. A legal auditor can skip files that do not to include any license, and concentrate on those with an unknown one.

- They report false positives. *ohcount* and *OSLC* use simple regular expressions that might be matched even if the intention of the licensor is the opposite. For example, we have encountered cases in which these systems report a file as using the General Public License because it was found in text such as: "This file is not licensed under the GPL". We presume a legal auditor will prefer to scan a file manually than to incorrectly assume it has a given license.

- FOSSology is slow. In our experience, FOSSology is more accurate than the other two (we perform a comparison further below that corroborates this fact), but runs one or two orders of magnitude slower. It requires also a complex setup, including a running database.

Barahona et al. measured different characteristics of the Debian distributions, but did not include licensing in their study [12].

2. LICENSING OF SOURCE CODE

The license of a source code file is typically found in a comment at the very beginning of the file. We refer to this region of the file as its **license statement**. A license statement typically contains four sections (not necessarily in this order) [5]: 1) a list of copyright owners, 2) a list of authors

(if different from the copyright owners), 3) the license or licenses that cover the file (see below for details), and 4) warranty and liability statements. The licenses in the licensing statement can be of two types:

by-inclusion: the text of the license is embedded in the file, examples are the BSD and the MIT families of licenses;

by-reference: the license statement indicates where the text of the license is can be found. This can be a file (in the case of the family of GPL licenses) or a URL (as the Apachev2, the MPLv1.1, and the CPL).

In order to understand the characteristics of license statements as they appear in FOSS, we have created a corpus of source code with approximately 30,000 files. This included applications such as Linux, Eclipse JDT, OpenBSD, FreeBSD, and Mozilla. We have analyzed the licensing statements of these files in a combination of manual and automatic methods, trying to determine the various ways in which different licenses are present in files. We also studied the Open Source Initiative (OSI) approved licenses, and the corpus of licenses of FOSSology. This exploratory study resulted in the identification of challenges of license identification, which are summarized in Table 2. They fall into three categories: 1) **Finding the license statement**: how many license are there in the file, where (within the file) they are located; 2) **Language related**: the grammar, spelling, and wording use in a license statement; and 3) **License customization**: the terms of the licenses are sometimes altered by their users. Each of these challenges is described below:

F1. License statements are usually mixed with other text. Frequently the first comments of a file contain text that is not part of the licensing statement, such as a description of the file, or a ChangeLog. One exception is the files in various Mozilla projects, in which well-defined delimiters mark the beginning and end of the licensing statement, simplifying their identification.

F2. Files might reference another file where the license is located. Frequently a file will contain a licensing statement such as *"For licensing information, see the file [...]"*, *"See [...] for licensing information"*, *"See [...] for license details"*, etc.. Sometimes the file is in the same directory, sometimes in a *"main"*, *"root"* or *"top"* directory. The name of the file varies: *"COPYING"*, *"LICENSE"*, *"README"*, *"copyright.txt"*, *"AUTHORS"*, *"COPYRIGHT_and_LICENSE"*, another source code file, etc. (For the sake of space, we use [...] to abbreviate sections of text in the examples.)

F3. Files might contain multiple licenses. It is not uncommon for files to contain more than one license. Such licenses can apply in a conjunction (the terms of all licenses apply) or a disjunction (the terms of only one license, at the choice of the licensor, applies; see disjunctive licensing pattern in [9]). Determining the number of licenses present in a file, and how they interact with each other, is not trivial.

L1. Licensing statements might contain spelling errors. This is not surprising. A very common mistake we found is words split by whitespace.

L2. A given license is referred in different ways. When a file is licensed by-reference, usually the creator of the license documents the recommended way to create the reference (such as the GPL family of licenses, the MPL v1.1 and the EPL v1.0). Unfortunately this is not always the case. For example, we discovered files that refer to the GPL as *"This [...] is licensed under the GPL"*, *"You may use [...] under the GNU public license if you so wish"*, *"[...] is provided under the provisions of the GPL"*, where [...] is replaced by *"software"*, *"library"*, *"program"* or the name of the product. We also found the variant: *"[...] provides this source code under the GPL License"* where [...] is replaced with the name of the copyright owner, or its author.

L3. Licensors change the spelling/grammar of a license. Sometimes users change the grammar or spelling. For example, they replace *"license"* with *"licence"*, *"it would be useful"* with *"it will be useful"*, *"developed by"* with *"written at"*, *"dealings in"* with *"dealings with"*, *"Redistribution"* with *"Redistributions"*. The changes are also in punctuation: they add or remove a comma or a semicolon. Grammar changes are particularly significant because they might have the effect of changing the meaning of the license (intended or unintended). It should be left to legal experts to determine if these changes to a license should be considered a different license or not.

C1. Several licenses must be customized when used. Licenses such as the BSD must be customized to include the name of the copyright owner of the software. For instance, the warranty statement of the BSD sentence should read *"THIS SOFTWARE IS PROVIDED BY <name> AS IS AND ANY EXPRESS [...]"* with the name of the copyright owner in place of <name>.

C2. Licensors modify, add or remove conditions to well known licenses. Frequently a well-known license is modified to create a new license. Two of the best known examples are the BSD3 license, and the Apache v.1.1. The BSD3 is derived from the original BSD4, by removing a clause (famously known as the advertisement clause). The Apache License v1.1 is composed of the BSD3 license plus four more sentences. The OpenSSL license is a derivative of the Apache License v1.1. Other licenses derived from the BSD licenses are the Zlib/libPng license and the Sleepycat License. In other instances, the modifications are more subtle. For example, we found many cases in which the MIT/X11 license was changed from *"Permission to use, copy, modify, distribute, and sell this software [...]"* to *"Permission to use, copy, modify and distribute this software [...]"* (dropping *sell*). Similarly, the BSD condition *"Redistributions of source code must retain [...]"* is changed to *"Redistributions of source code or documentation must retain [...]"* (adding *or documentation*). Some licenses by-reference, such as the GPL and the MPL allow the licensor to modify the license using an addendum (this avoids the need to create variants of the license). These addenda are usually known as exceptions (they create exceptions to the conditions of the license–see Exception Pattern in [9]). These exceptions usually follow the reference to the license. For example, parsers generated with Bison contain files that are licensed under the GPLv2+ with a special addendum that allows their use under any license: *"As a special exception, when this file is copied by Bison into a Bison output file, you may use that output file without restriction."*

C3. Licensors modify licenses for various intents. By-inclusion licenses are prone to being modified by its users, willingly or unwillingly. For example, we have found variants of the BSD endorsement clause such as: *"<copyright owner names> may not be used to endorse [...]"*, *"Neither*

Type.	Challenge
Finding the license statement	*F1.* License statements are usually mixed with other text *F2.* Files might reference another file where the license is located *F3.* Files might contain multiple licenses
Language related	*L1.* Licensing statements contain spelling errors *L2.* A given license is referred in different ways *L3.* Licensors change the spelling/grammar of the license statement
License customization	*C1.* Several licenses must be customized when used *C2.* Licensors modify, add or remove conditions to well known licenses *C3.* Licensors modify licenses for various intents

Table 2: Major challenges of license identification.

the names of the authors may not be used to endorse [...]", and *"The name of the author may not be used to endorse [...]"*. Similarly, we found in some cases *"without prior written permission"* and in others *"without specific prior written permission"*. In the MIT/X11 we discovered the words *developer*, *author* and *copyright owner* used interchangeably; we also found *"software and associated documentation files"* replaced with *"source file"*; and in other files *"The above copyright notice [...] shall"* replaced with *"The above copyright notice [...] (including the next paragraph) shall [...]"*.

2.1 An algorithm for license identification

Based on our analysis of the corpus of source code files and the challenges described before, we developed a license identification algorithm. Its goals are to properly identify each of the licenses in a file (license name and version) and to sacrifice recall (being able to determine the license of a file) for the sake of precision (making as few mistakes as possible), and to run efficiently.

It works by extracting the license statement from the file, it then breaks it apart into textual sentences, and proceeds to find a match for each of them individually; the list of these matched sentences is analyzed to determine if it contains one or more licenses. This method requires a knowledge-base with three sets of information: 1) filtering keywords $\mathcal{K}$, 2) sets of equivalence phrases $\mathcal{E}$, 3) known sentence-token expressions $\mathcal{T}$, 4) license rules $\mathcal{R}$. These sets were created during the study of our corpus and are described below.

Set of filtering keywords $\mathcal{K}$: One of the challenges of license identification is discriminating between those sentences that are part of the license statement and those that are not relevant (see challenges F1, and F3 above). Any sentence that does not match at least one keyword is not expected to contribute to the license of the file. Keywords can be composed of one or more words. Examples of filtering keywords are *license*, *conditions*, *disclaimer*, *written permission*. We have identified 82 such keywords.

Set of sets of equivalence phrases $\mathcal{E}$: To deal with language related challenges (L2 and L3) we have created sets of equivalent phrases. Any phrase that is in a set is considered semantically equivalent to any other in the same set. Examples of these equivalence phases sets are: *(at your option) any later version*, *any later version* or *any greater version*; *distributable*, *licensed*, *released* or *made available*. These sets should be created carefully: replacing one phrase with another in the same equivalence set should not change the meaning of the sentence. For example, some might argue that "you may" is not always equivalent to "you can". This includes punctuation (such as different types of matching quotes " ", ‘ ’, ’ ’, ” ”. We have created 12 such sets.

Set of known sentence-token expressions $\mathcal{T}$: We use the term **sentence-token** to refer to a sentence of a known license. A license (both by-inclusion or by-reference) is a sequence of sentence-tokens. Sentence-tokens are generalized using one or more regular expressions (see challenges C1, C2, and C3). We refer to the pair ⟨sentence-token, regular expression⟩ as a sentence-token expression. The objective of this set is to translate each sentence found in the licensing statement into a sentence of a known license (a sentence-token). We have identified 427 sentence-token expressions. For example, two of the sentence-token expressions matching variants of the sentence-token *"SeeFile"* (the license is inside a given file name) are:

SeeFile,`^For the licensing terms see the file ([^,;]+)$`
SeeFile,`^See ([^,;]+) for license$`

Set of license rules $\mathcal{R}$: Each license corresponds to a sequence of one or more sentence-tokens (which we call a **license rule**) plus a set of non-critical sentence tokens (which we call its **associate sentence-tokens**). Most by-inclusion licenses require matching two or more consecutive sentence-tokens. In contrast, by-reference licenses usually require only one sentence-token to be matched (indicating the intention to use such license). We have identified 126 licenses rules to match 112 licenses. For example, the BSD2 license rule is the sequence of 5 sentence-tokens, $\langle BSDpre, BSDcondSource, BSDcondBinary, BSDasIs, BSDWarr\rangle$; in other words, these 5 sentence-tokens should appear contiguously and in the same order for the file to be considered licensed under the BSD2. Each rule also contains a list of sentence-tokens that usually appear with the license, but are not required for the file to have such license (its associate sentence-tokens); for example, a file licensed under a GPL license will also contain the typical GPL warranty and liability sentences (as recommended by the FSF).

Our algorithm for license identification is divided into six steps, detailed below; they are illustrated in Figure 1.

1. **License statement extraction:** The comments at the header of the file are extracted. If no comment extractor for the programming language exists, we simply extract the first 1,000 lines of text (the longest license statement we have found is 700 lines long).

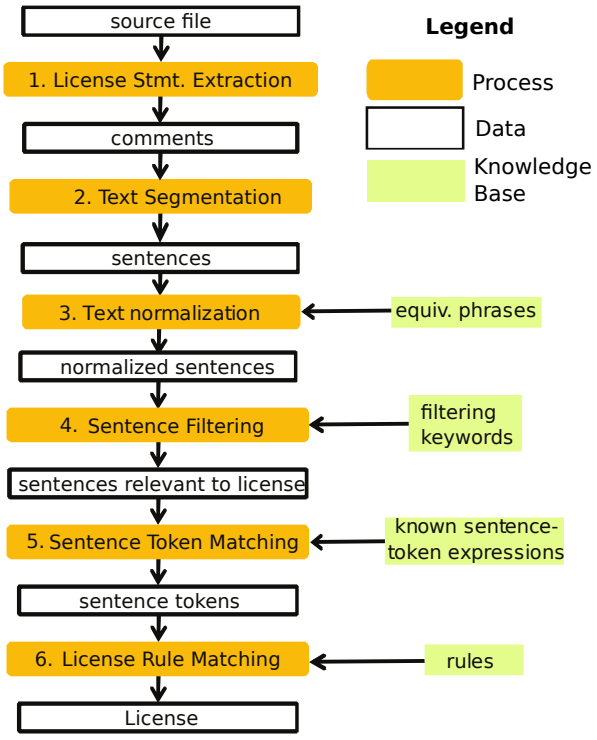

Figure 1: Diagram depicting the main steps of the algorithm of license identification.

2. **Text segmentation:** Using a text segmentation algorithm, the comments are converted into a sequence of statements S. Our implementation is based upon [3] with further modification to remove source code comments and special characters used as prefixes in each line (such as # and |).

3. **Equivalent phrase substitution:** Each sentence is scanned for occurrences of phrases in $\mathcal{E}$, and if found, replaced with a normalized version of the phrase.

4. **Sentence filtering:** S is split into two sequences, S_+ and S_-. A statement is part of S_+ if it contains at least one legal keyword $k \in \mathcal{K}$, otherwise it is part of S_-. The relative order of the sentences in S is preserved in S_+ and S_-. If S_+ is empty, the file does not have a license, and we label the source file $NONE$. S_+ contains the sequence of all sentences that are expected to contribute to the licensing of the file.

5. **Sentence-token matching:** $\forall s_i \in S_+$, we find its corresponding sentence-token $t_i \in \mathcal{T}$ such that the regular expression corresponding to t_i matches s_i. If no such t_i exists, we use the sentence-token $UNKNOWN$ instead. Some regular expressions contain variable sections (for instance, the BSD license requires the name of the copyright owner to be included as part of some of its sentences). Hence each sentence in S_+ is equivalent to a pair consisting of the sentence-token plus its list of k optional parameters $\langle t_i, < p_1, ..., p_k > \rangle$ for a value of $k \geq 0$. The result of this step is a sequence of sentence-tokens (with parameters), each equivalent to their corresponding sentence in S_+. We call this sequence M.

6. **License rule matching:** $\forall r \in \mathcal{R}$, we try to match r to M. If r is contained in M, we output the license l that corresponds to r. The sentence-tokens of M that match r and any sentence-token that matches an associated sentence-tokens of r are marked as matched. This step stops when all sentence-tokens in M are matched or all rules in $\mathcal{R}$ are exhausted (see challenge F3).

Result: The license of the file is the list of licenses matched in the previous step. If no license is matched, the file is considered to have an $UNKNOWN$ license. From M a sequence of sentence-tokens that were not matched is built. We will refer to it as the **list of unmatched sentences**.

It is possible that the algorithm identifies a valid license, and its list of unmatched sentences still contain known and unknown sentence-tokens (in other words, part of the license statement of the file has been identified, but another part has not). This poses a risk that the file might contain one or more licenses that have not being identified (see F3), and therefore it is recommended that these lists be inspected manually. The list of unmatched sentences of all files can be aggregated and inspected all at once (instead of having to inspect each file individually).

2.2 Example

We illustrate our method with this license statement:

```
/* Copyright (C) 2006 Apple Computer, Inc.  All rights reserved.
* Copyright (C) 2006 Michael Emmel mike.emmel@gmail.com
* Copyright (C) 2007, 2008 Alp Toker <alp@atoker.com>
*
* Redistribution and use in source and binary forms, with or
* without modification, are permitted provided that the following
* conditions are met:
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above
*    copyright notice, this list of conditions and the following
*     disclaimer in the documentation and/or other materials
*     provided with the distribution.
*
* THIS SOFTWARE IS PROVIDED BY APPLE COMPUTER, INC. ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
* TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
* PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL APPLE
* COMPUTER, INC. OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
* BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
* THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE. */
```

This license statement is segmented into 7 sentences. Sentence 6 is modified to replace “ with <QUOTES> (its preferred equivalent phrase). The number in front of each line identifies its order in the original licensing statement:

```
1 Copyright (C) 2006 Apple Computer, Inc. All rights reserved.
2 Copyright (C) 2006 Michael Emmel mike.emmel@gmail.com Copyri[...]
3 Redistribution and use in source and binary forms, with or w[...]
4 Redistributions of source code must retain the above copyrig[...]
5 Redistributions in binary form must reproduce the above copy[...]
6 THIS SOFTWARE IS PROVIDED BY APPLE COMPUTER, INC. <QUOTES>AS[...]
7 IN NO EVENT SHALL APPLE COMPUTER, INC. OR CONTRIBUTORS BE LI[...]
```

The second sentence is filtered out (it contains no filtering keywords). The remaining sequence of sentences is translated to the following sequence of sentence-tokens and their parameters:

```
1 <AllRights, <Copyright (C) 2006 Apple Computer, Inc.>>
3 <BSDpre, <:>>
4 <BSDcondSource, <above>>
5 <BSDcondBinary, <>>
6 <BSDasIs, <APPLE COMPUTER, INC.>>
7 <BSDWarr, <APPLE COMPUTER, INC. OR CONTRIBUTORS>>
```

Each line is composed of the name of a sentence-token, followed by a list of instantiated parameters. For example, line 1 has sentence-token *AllRights* with a parameter of the leading string *"Copyright . . . "*. Line 2 *BSDpre* is sometimes followed by a colon. Line 3 *BSDcondSource* includes the parameter *"above"* because we have found two variants of this license, one using *"above copyright notice"* and another using only *"copyright notice"* (this variant is context dependent and cannot be an equivalence phrase). The purpose of these parameters is to simplify the identification of variants of the license, and to inform the user of how they are matched, as sometimes there might be errors in them (e.g. the copyright owner being different in the *BSDasIs* and *BSDWarr* sentences, or incorrect) or a rule might have a regular expression that is too greedy and match more than the intended text. In this example, the name of the copyright owner (*Apple Computer, Inc.*) is consistent in the three matched rules.

This list matches the BSD2 rule: $\langle BSDpre, BSDcondSource, BSDcondBinary, BSDasIs, BSDWarr\rangle$ and $AllRights$ is marked as an associate sentence-token of the BSD2. For this license statement, no unmatched or unknown sentences exist. Hence, this file contains BSD2 license only.

2.3 Ninka

Using this algorithm, we have developed a license identification tool called `Ninka`. Ninka is open source licensed under the General Public License version 3[1]. It consists of approximately 1,500 lines of Perl, 82 filtering keywords, 12 equivalence phrases, 427 sentence-token expressions, and 126 license rules. It is capable of identifying 112 licenses.

3. EVALUATION

We conducted a comparison experiment to evaluate `Ninka` against other existing tools: FOSSology version 1.0.0[11], ohcount version 3.0.0rc and OSLC 3.0. FOSSology identifies licenses using the Binary Symbolic Alignment Matrix (bSAM) pattern matching algorithm, matching the text files against license templates contained in a database. FOSSology is able to detect a wide variety of specific licenses (version 1.1 is capable of recognizing 360 different ones). Its main disadvantage is its speed (it takes from few seconds to few minutes to identify the license of a file). Ohcount[2] is developed by Ohloh, an open source directory. Ohcount uses simple regular expressions to match licenses. OSLC[3] finds exact or partial matches against the licenses stored in its database using an algorithm based on a technique for isolating textual differences [13].

[1]Ninka is currently available by request and will be made publicly available after this paper is published.
[2]http://ohcount.sourceforge.net/
[3]http://forge.ow2.org/projects/oslcv3/

3.1 Setting

In order to guarantee fairness, the evaluation was performed by one of the authors independently from the maintainer of `Ninka`. We did not want our tool to be customized (consciously or unconsciously) to the files in this study and consequently improve `Ninka`'s accuracy.

We selected 250 source code files. First, we randomly selected 250 applications from Debian 5.0.2. We then randomly selected one file from each. These 250 files form the set N. Our goal was get as many different types of licenses and licensing statements as possible. If we consider that our sample comes from 0.8 million source code files in Debian 5.0.2, our study will provide us with a 95% confidence level of an error of $\pm 6.2\%$ that we can correctly identify the licenses in the source code files of Debian 5.0.2.

The evaluation was conducted using the following procedure: First, for each source file in N, its license (or licenses) were manually identified. Second, each tool was used to identify the licenses in each file. We used the following commands: for FOSSology, `fosslic`, for ohloh `ohcount -l`, and for OSLC `java -jar oslc.jar -d`.

Each of these tools has a different idiosyncrasy in the way it reports its results. For example, the names used for each of the licenses might be different (this is particularly true for by-inclusion licenses) hence we inspected the documentation of each tool trying to determine if the license name it uses corresponds to the same license we found. OSLC displays the percentage of match of a license; we disregard this number and considered each of the licenses it outputs as being identified.

Because files without a license are fairly common (28.4% in our sample did not have a license) we assume that, if the tool does not output any license, the tool did not find one, and will be considered correct if there is no license in the corresponding file. Neither ohcount nor OSLC report when they find an unknown license. For FOSSology we consider the output *"Phrase"* as an unknown license. `Ninka` explicitly states if there is a license that it cannot recognize. The data used in this experiment and the results of each tool can be found at `http://turingmachine.org/ninkaData/`. We divided the results of each tool into 3 sets:

- C **Correct license name and version.** The tool correctly identifies each of the licenses in the file, including their version (where applicable, i.e. the official name of the license contains a version number, such as the GPL, or the CPL; we consider the BSD4, BSD3, BSD2 different versions of the same family). If the file contains no license, the tool outputs nothing or that there is NO license in the file. If the file contains more than one license, each of them should be correctly identified.
- I **Incorrect.** The tool identifies a license but it is incorrect. We also consider it an incorrect identification when the tool outputs multiple licenses and at least one is incorrect.
- U **Unknown.** The tool explicitly states that it has found a license and it does not recognize it.

These sets are mutually disjoint, and $N = C \cup I \cup U$. Finally, we calculate the recall and precision of the results of each tool as follows:

$$Recall = \frac{|C|}{|C| + |U|} \qquad Precision = \frac{|C|}{|C| + |I|}$$

$$\text{F-measure} = \frac{2 * |C|}{2 * |C| + |I| + |U|}$$

Intuitively *Recall* measures how good is the tool at discovering that there is a license in the file. Because neither ohcount nor OSLC support reporting unknown licenses their recall is 100%. In contrast, `Ninka` and FOSSology are capable of reporting when a license is found, but is not identified. We believe this is an important feature. This information is valuable to the person doing the license analysis: it is better to know that a file might contain an unidentified license, rather than to think that it contains no license, or worse, an incorrect one (i.e., if a file contains license A it is better to say *"I don't know"* than to err *"the file has no license"* or *"the file has license B"*).

Precision measures how good is the tool at identifying the correct license and version. Comparing each tool simply in terms of their precision or recall is insufficient. Therefore we use the F-measure to provide a more balanced comparison of the trade-off between precision and recall.

3.2 Results

Table 3 shows the recall, precision and F-measure of the results of each of the tools in our study, and the time it took each to analyze the set of files.

Table 3: Results of the evaluation of the tool. Best result in each category is depicted in bold.

	`Ninka`	FOSSo.	ohcount	OSLC
$\|C\|$	200	137	83	57
$\|I\|$	7	112	167	193
$\|U\|$	43	1	0	0
Recall	82.3%	99.2%	**100.0**%	**100.0**%
Precision	**96.6**%	55.0%	33.2%	29.5%
F-measure	**0.889**	0.708	0.498	0.371
Execution Time	**22s**	923 s	27s	372s

We can make several observations about the results. `Ninka` sacrifices recall for the sake of precision. It prefers to report the license as unknown rather than making a mistake. In fact, of the 6 files incorrectly identified by `Ninka`, in three it correctly identified one of the licenses in the file, but no all of them; and in three it reported the file as having no license when it did contain one. `Ninka` shows the highest F-measure in these tools. This is reassuring: its low recall (`Ninka`'s recall is the lowest of the four) is offset by its high precision.

FOSSology and OSLC are significantly slower than the rest (30 and 10 times longer than ohcount or `Ninka`, respectively), and `Ninka` is slightly faster than ohcount. We believe this is because `Ninka` only analyzes the first lines of each file.

In conclusion, our study shows that we have achieved the main goals set for `Ninka`: it has the highest precision of all the tools in the study, and it does it efficiently (it was the fastest tool in the study).

4. EMPIRICAL STUDY

To demonstrate the usefulness of `Ninka` we performed a study of the licenses found in the files of a large open source distribution (Debian 5.0.2). More specifically, we have performed an empirical study to answer the following research questions:

RQ1: What are the licenses used in FOSS? No empirical study has been performed to identify the licenses of source code files in a large collection of FOSS applications. Our goal is to identify what licenses are used and how frequently.

RQ2: When present, what types of errors do license statements have? Intuitively we would expect license statements to contain errors. Our goal is to attempt to identify them and explore how the occur.

To address RQ1, we identified the licenses of the source files of Debian 5.0.2 using `Ninka` and evaluated a) how frequently each license is used; b) if there is a relationship between license and programming languages, and c) if smaller files are more likely not to have a license.

For RQ2 we looked for inconsistencies in licenses. We concentrated on applications where most of the files used the same license, and only a handful had a different one. We then manually analyzed the differences, checked available documentation for clarification, and contacted the authors of the software to verify our findings.

4.1 Setting

We downloaded the source code of Debian 5.0.2 from Debian's official repository[4] We analyzed all files written in Java (.java), C (.c), C++ (.cpp, .cxx, and .cc), Lisp (.el and .jl), Perl (.pm and .pl), and Python (.py); according to [12] these are five of the six most frequently found programming languages used in Debian—the other is shell programming. In Debian 5.0.2 we analyzed 794,622 files from 11,101 applications (also known as source packages in Debian's nomenclature). `Ninka` could not identify a license in 15.9% of files. These numbers are consistent with our empirical evaluation of `Ninka` in Section 3.2. `Ninka` was able to identify the license of all files in 31.6% of all applications.

4.2 RQ1: What are the licenses used in FOSS?

Due to space constraints we only present some results. We invite the reader to download the list of files and their licenses from `http://turingmachine.org/ninkaData/`.

4.2.1 What licenses are used by software in Debian

Table 4 shows the most commonly found licenses in Debian 5.0.2 files. The files with no license statement are the most frequent (NONE). The GPL and LGPL licenses, in all their versions, appear prominently in the list: they are widely used by many files, in many different applications. On the other hand, some licenses are used in few applications but these applications contain many files, e.g. the disjunctive license *"CDDLv1 or GPLv2"* it is only used in two applications (one of them *netbeans* with 28,523 files). In comparison, Table 5 shows the number of applications in which a given license is present in at least one of its file. The GPLv2+ is the most common in both tables.

With regard to the distribution of licenses per application, the number of licenses used in each application tends to be very small. The median number of different licenses

[4] `ftp://ftp.debian.org/debian/dists/Debian5.0.2/`

License	Files	Perc.	Aps.
NONE	210147	31.5%	8241
GPLv2+	147535	22.1%	5486
LesserGPLv2.1+	42692	6.4%	767
CDDLv1orGPLv2	37623	5.6%	2
SeeFile	31685	4.7%	1252
Apachev2	25023	3.7%	151
LibraryGPLv2+	23705	3.6%	1150
GPLv2	18930	2.8%	582
GPLv2 or LGPLv2.1 or MPLv1.1	18183	2.7%	30
GPLv3+	16767	2.5%	224
BSD3	12394	1.9%	646
MITX11	10237	1.5%	601
LesserGPLv2+	7783	1.2%	470
BoostV1	7348	1.1%	13
GPLv2+,LinkException	6029	0.9%	11
BSD2	3877	0.6%	255
LesserGPLv2.1	3496	0.5%	108
LibraryGPLv2	3283	0.5%	100
SameAsPerl	3275	0.5%	791
GPLnoVersion	3047	0.5%	334
MITX11 variant	2521	0.4%	57

Table 4: Most common licenses per number of files where they occur in Debian 5.0.2. The last column is the number of applications in which they are found.

detected per project is 1. 5186 (47.2%) had only one detected license, and 1956 applications (17.6%) had two. The maximum number of licenses in one application is large; e.g., 40 different licenses in *linux*, 31 in *vnc4*, and 26 in *texlive*.

These numbers could be skewed because `Ninka` did not detect the licenses of all the files. To avoid this potential problem we studied applications in which `Ninka` resolved all their files: 3506 (31.8% of the total) (i.e. for each file in it, `Ninka` either identified its license or that it contained no license). 2183 of these applications used only one license. Table 6 shows the distribution of licenses in these applications. Another 557 applications used two licenses only.

License	Aps.	Prop
NONE	8241	74.2%
GPLv2+	5486	49.4%
SeeFile	1252	11.3%
LibraryGPLv2+	1150	10.4%
SameAsPerl	791	7.1%
LesserGPLv2.1+	767	6.9%
MITX11	601	5.4%
BSD3	646	5.8%
GPLv2	582	5.2%
LesserGPLv2+	470	4.2%
GPLnoVersion	334	3.0%
BSD2	255	2.3%
publicDomain	244	2.2%

Table 5: Most common licenses per number of applications in which they appear at least once in Debian 5.0.2.

4.2.2 Do different programming languages use different licenses?

License	Aps.	Perc.
GPLv2+	1265	57.9%
SameAsPerl	369	16.9%
MITX11	86	3.9%
BSD3	83	3.8%
SeeFile	63	2.9%
Apachev2	55	2.5%
LesserGPLv2.1+	44	2.0%
GPLv2	38	1.7%
BSD2	24	1.1%
LibraryGPLv2+	16	0.7%

Table 6: Most common licenses used by applications that used only one license, and every file was resolved (it either had no license or it was known) in Debian 5.0.2. There were 3506 such applications (31.8% of all applications).

Table 7 shows the distribution of licenses grouped by programming language (based upon the extension of the file). This table shows the results for Java, C, C++, and Perl.

With respect to Java, as it can be observed, two applications (*netbeans* and *glassfish*) contribute all the files that appear with the *"CDDLv1 or GPLv2"* (Java's most used license). It is also significant that other licenses appear before the GPLv2+.

C programs, on the other hand, are more likely to use the FSF licenses. C++ programs were found to frequently use the triple license GPLv2, LGPLv2.1 or MPLv1.1, promoted by the Mozilla Foundation; the large number of files that compose Boost pushes the number of files under its license into this list.

Perl is peculiar because it uses an indirect license: *"Same License as Perl"*. This is the default license statement included in the skeletons of Perl modules automatically created by Perl, and this is likely the reason behind its frequency[5]. The table also shows Perl files are more likely not to have a license (67%, more than twice of any of the other three languages).

4.2.3 Does size matter?

During the development of `Ninka` we analyzed various FOSS applications to test the efficacy of our tool. One of these applications had a significant proportion of files without a license. We contacted its main developer, informing him of this. He replied that he did not think it was necessary to place a license on very small files (usually test files).

Are smaller files more likely not to have a license? The median size of files in Debian 5.0.2 is 4633 [2041,11863] bytes; the median size of files without a license is 2137 [708,6857] bytes, while the ones with a license is 5488 [2868,13217]. Considering that the median size of the license statement of a file is 1005 bytes, the median size of files with a license was still more than 2 kilobytes longer.A Mann-Whitney test confirms that these differences are significant ($p < 0.0001$). However, we did not find a correlation between the size of

[5]This type of license makes it easy to relicense these modules if and when Perl changes it license[9]. Perl is expected to change its license to the Artistic v2 when version 6 is released; it is currently using the disjunctive license between the Artistic v1 and the GPLv1+.

License	Files	Perc.	Aps.
Java (.java)			
CDDLv1 or GPLv2	37562	25.43%	2
NONE	25371	17.17%	344
LesserGPLv2.1+	22834	15.46%	61
Apachev2	21535	14.58%	100
GPLv2+	10679	7.23%	72
GPLv2+,LinkException	5888	3.99%	8
c (.c)			
NONE	85261	32.25%	3726
GPLv2+	66654	25.21%	2718
LesserGPLv2.1+	14981	5.67%	513
SeeFile	14926	5.65%	357
LibraryGPLv2+	12440	4.70%	795
GPLv3+	10191	3.85%	133
C++ (.cpp, .cxx, .cc, and c++)			
GPLv2+	47528	31.63%	999
NONE	36661	24.40%	1171
GPLv2 or LGPLv2.1 or MPLv1.1	12169	8.10%	15
SeeFile	9704	6.46%	95
LibraryGPLv2+	8854	5.89%	189
boostV1	6554	4.36%	11
Perl (.pm and .pl)			
NONE	18227	66.73%	2200
GPLv2+	3079	11.27%	383
SameAsPerl	2651	9.71%	584
LibraryGPLv2+	681	2.49%	233
SeeFile	556	2.04%	180
GPLv2 or LGPLv2.1 or MPLv1.1	293	1.07%	9

Table 7: Most frequent licenses used by each programming language in Debian 5.0.2

the file and the existence of a license in it ($\rho = 0.1347$).

4.3 RQ2: When present, what types of errors do license statements have?

During the development and testing of `Ninka` we observed several potential problems in the licensing of various applications that we analyzed. These are summarized below:

Application #1. Files without a license We contacted the author of this application informing him that one of its files did not contain a license (the rest contained the GPLv3+). He acknowledged that this file should not have been distributed with the software and removed it.

Application #2. Cutting & pasting the wrong license statement In this application we discovered a license inconsistency. The application, according to its web site, was licensed under the GPLv2. 169 of its files were licensed under the GPLv2+, 3 under the GPLv3+, one in the public domain, and 26 didn't have a license. The GPLv3+ files created a licensing conflict: a file licensed under the GPLv3+ cannot be linked into a system under the GPLv2. We emailed its mailing list asking for a clarification. The project leader quickly replied: *"Those files did not have a copyright header and no license at all, I've copied a license header into them, but did not see that it was the wrong one, sorry."* He also clarified that files without a license statement are licensed under the GPLv2+.

Application #3. Inconsistent license clauses This application is licensed under an MIT-style license. 3 files out of 37 contained different liability and warranty clauses than the rest. When we contacted the author of the application, he acknowledged that this was a problem and would be fixed in the next release.

Application #4. Incorrect name of the license According to the Web site of this application, it is licensed under the Lesser GPL v2.1. Its README.txt file asserts it is licensed under the Library GPLv2+. `Ninka` identified 8 files under the Lesser GPLv2+, but this license does not exist. The successor of the Library GPL version 2 is the Lesser GPL version 2.1 (there was never a version 2.0 of this license). The rest of the files correctly referred to the Library GPLv2+. When we contacted the author of the application, he acknowledged the problem, claiming the licenses of all files will be changed to Lesser GPLv 2.1.

Application #5. License statements can only be edited by their copyright owners This application included a file under the Lesser GPLv2+ (as mentioned above, this version of the license does not exist). When we contacted the project leader he informed us that this was a known problem, and there was a bug report already filled about it. Unfortunately this file had come from a different application, and hence because they were not the copyright owner they could not change its license statement; instead they were trying to find its copyright owner to perform such change

In summary, these issues highlight the need for license maintenance features in software development environments that support:

- Editing of the license statements.
- Verifying the validity of the license statements.
- Summarizing licenses in source code files.
- Tracking of copyright owners.

5. THREATS TO VALIDITY

With respect to our evaluation of `Ninka` and the other tools, we addressed *construction validity* by randomly choosing files from many different applications. We addressed *conclusion validity* by performing a statistical test that indicates the level of confidence of the results.

Our study of licensing in FOSS is exploratory and its main threat to *construction validity* is how reliable our license detection is. Our empirical evaluation of `Ninka` has been design to quantify it. In this evaluation `Ninka` had a 96.6% precision for files in which it identified a license, hence we are confident of the quality of our analysis of the complete Debian distributions. There is, however, the risk that the unknown licences might change some of the results. This potential impact would be small because only 15.9% of files in Debian 5.0.2 contained a license that was not identified. We inspected many of these files and, in most cases, they contained variants of other known licenses that were not recognizable by `Ninka`. Furthermore, we restricted some of our analysis to applications for which `Ninka` identified a license (or lack of one) in each of their files. With respect to *external validity*, we believe that our results are representative to FOSS in general because a) we analyzed a very large number of files and applications, and b) Debian serves as a proxy of active (or at least used) FOSS applications. We have also contacted developers (where applicable) to make sure that our interpretations were not erroneous.

With respect to *reliability validity*, our study can be replicated: first, we have used a well defined collection of source code files that is available to anybody (Debian distributions); second, we are making `Ninka` available under an open source license; and third, the data and results or our experiments are available for others to study at `http://turingmachine.org/ninkaData/`.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a method of license identification for source code files. This method is based on matching each of the sentences of the license statement of the file to the sentences of known licenses. Based on this method we implemented `Ninka`. We empirically evaluated `Ninka` and other similar tools and found that `Ninka` is better and faster at identifying the license name and version than these tools, at the cost of lower recall.

We have performed an empirical study of a large collection of free and open source applications. We discovered, for example, that the GPLv2+ is the most commonly used license in terms of number of files and number of applications that use it.

We also discovered that licensing statements are prone to errors. The integration of license verification tools into development environments will alleviate this problem.

With respect to future work, the identification of licenses of files is only one step in licensing analysis of components. It is also necessary to analyze the interactions between different files of a system to determine the resulting license of a component, and, at a higher level, how the interaction of different components affect the overall license of a system that uses them.

Acknowledgements

We would like the thank the reviewers of an early draft of this paper. This work has been supported by the Invitation Fellowship Program of the Japan Society for Promotion and Science, the Grant-in-Aid for Scientific Research (A) (No.21240002), the Stage Project, the Development of Next Generation IT Infrastructure of Mext Japan, and the National Science and Engineering Research Council of Canada.

7. REFERENCES

[1] T. A. Alspaugh, H. U. Asuncion, and W. Scacchi. Analyzing software licenses in open architecture software systems. In *FLOSS '09: Proc. Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*, pages 54–57, 2009.

[2] A. Capiluppi, P. Lago, and M. Morisio. Characteristics of open source projects. *Software Maintenance and Reengineering, European Conference on*, 0:317, 2003.

[3] P. Claugh. A Perl program for sentence splitting using rules. `http://ir.shef.ac.uk/cloughie/software.html`, April 2001.

[4] J. Colazo and Y. Fang. Impact of license choice on open source software development activity. *J. Am. Soc. Inf. Sci. Technol.*, 60(5):997–1011, 2009.

[5] M. Di Penta and D. M. German. Who are source code contributors and how do they change? In *"Proc. 16th Working Conference on Reverse Engineering WCRE'09"*, pages 11–20, Oct 2009.

[6] M. Di Penta, D. M. German, Y.-G. Guéhéneuc, and G. Antoniol. An exploratory study of the evolution of software licensing. In *Proc. of the 32rd Int. Conf. on Software Engineering (ICSE 2010)*, 2010. To appear.

[7] Free Software Foundation. Frequently Asked Questions about the GNU Licenses. `http://www.fsf.org/licensing/licenses/gpl-faq.html`. Accessed Feb. 2009.

[8] D. M. German, M. Di Penta, and J. Davies. Understanding and Auditing the Licensing of Open Source Software Distributions, 2010. Under review.

[9] D. M. German and A. E. Hassan. License integration patterns: Addressing license mismatches in component-based development. In *Proc. 31st Int. Conf. on Soft. Eng., ICSE*, pages 188–198, 2009.

[10] D. M. German, M. D. Penta, Y. Gueheneuc, and G. Antoniol. Code siblings: Technical and legal implications of copying code between applications. In *Proceedings of the International Working Conference in Mining Software Repositories*, pages 81–90, 2009.

[11] R. Gobeille. The FOSSology project. In *MSR '08: Proceedings of the 2008 International Conference on Mining Software Repositories*, pages 47–50, 2008.

[12] J. M. Gonzalez-Barahona, G. Robles, M. Michlmayr, J. J. Amor, and D. M. German. Macro-level software evolution: a case study of a large software compilation. *Journal of Empirical Software Engineering*, 14(3):262–285, 2009 2009.

[13] P. Heckel. A technique for isolating differences between files. *Commun. ACM*, 21(4):264–268, 1978.

[14] J. Li, R. Conradi, C. Bunse, M. Torchiano, O. Slyngstad, and M. Morisio. Development with off-the-shelf components: 10 facts. *Software, IEEE*, 26(2):80–87, 2009.

[15] Nokia Corp. About PySide. `http://www.pyside.org/about/`, 2009. Acc. Sept. 2009.

[16] M. D. Penta, D. M. German, and G. Antoniol. Identifying Licensing of Jar Archives using a Code-Search Approach. In *International Working Conference on Mining Software Repositories (MSR 2010)*, May 2010. To appear.

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

[18] C. Ruffin and C. Ebert. Using open source software in product development: a primer. *IEEE Software*, 21(1):82–86, 2004.

[19] W. Scacchi. Free/open source software development. In *ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, pages 459–468, New York, NY, USA, 2007. ACM.

[20] C. Subramaniam, R. Sen, and M. L. Nelson. Determinants of open source software project success: A longitudinal study. *Decision Support Systems*, 46(2):576–585, 2009.

[21] T. Tuunanen, J. Koskinen, and T. Kärkkäinen. Automated software license analysis. *Automated Software Engg.*, 16(3-4):455–490, 2009.