A Qualitative Study on the Adoption of Copyright Assignment Agreements (CAA) and Copyright License Agreements (CLA) within Selected FOSS Projects

by **Sylvia F. Jakob**, LL.B.(Hons.), Dipl. L.P., LL.M., Solicitor (n.p), Research Assistant, Institute for Legal Informatics, Hannover

Abstract: Open source software projects are multi-collaborative works incorporating the contributions of numerous developers who, in spite of publishing their code under a public license such as GPL, Apache or BSD, retain the copyright in their contributions. Having multiple copyright-owners can make the steering of a project difficult, if not impossible, as there is no ultimate authority able to take decisions relating to the maintenance and use of the project. This predicament can be remedied by centring the dispersed copyrights in a single authority via contributor agreements. Whether to introduce contributor agreements, and if so in which form, is a pressing question for many emerging, but also for established projects. The current paper provides an insight into the ethos of different projects and their reason for adopting or rejecting particular contributor agreements. It further examines the exact set-up of the contributor agreements used and concludes that smart drafting can blur the difference between CAAs and CLAs to a considerable extent, manoeuvring them into a legal grey area. To avoid costly litigation to test the legal enforceability of individual clauses, this paper proposes the establishment of an international committee comprised of developers, product managers and lawyers interested in finding a common terminology that may serve as a foundation for every contributor agreement.

Keywords: Free and Open Source Software (FOSS), Public Licenses, Outbound Licenses, Inbound Licenses, Contributor Agreements, Contributor Assignment Agreements (CAA), Contributor License Agreements (CLA)

Recommended citation: Sylvia F. Jakob, A Qualitative Study on the Adoption of Copyright Assignment Agreements (CAA) and Copyright License Agreements (CLA) within Selected FOSS Projects, 5 (2014) JIPITEC 105, para. 1.

A. Introduction

1 Open source software projects are multi-collaborative works incorporating the contributions of numerous developers who, in spite of publishing their code under a public license such as GPL, Apache or BSD, retain the copyright in their contributions. The public license, also referred to as an "outbound license", regulates the usage rights granted by the developer to the outside world. It ensures that the code can be used by virtually everyone having an interest in doing so as long as the user follows the terms of the outbound license.

2 However, having multiple copyright-owners can make the steering of a project difficult, if not impossible, as there is no ultimate authority able to take decisions relating to the maintenance and use of the project. This predicament can be remedied by centring the dispersed copyrights in a single authority via contributor agreements, also referred to as "inbound licenses" because they regulate the relationship of the developer with a particular organizational entity.

3 In recent years many FOSS projects have incorporated as non-profit organizations1 and many corporations have begun to release protected code under open source licenses to harness the wisdom of the crowd.2 Many of these organizations require their contributors to sign a contributor agreement, either in the form of a Copyright Assignment Agreement (CAA), whereby the developer transfers and abandons his intellectual property rights in the contribution for the benefit of a project's administration, or a Contributor License Agreement (CLA),3 whereby the developer is only required to grant usage rights. Some projects, in turn, continue to follow the notion of "outbound" equals "inbound",4 arguing that a public license sufficed and no intellectual property management within the project was necessary.5

4 To date no comprehensive, legal study6 has been conducted asking which kind of projects use CAAs, which CLAs, and which forego the management of intellectual property of contributions entirely. The existence of a pattern would be particularly interesting for new projects, as many find it difficult to determine into which end of the spectrum they fall, whether to use a contributor agreement, and if so, how to draft it, or where appropriate, to refrain from using a contributor agreement from the outset.

5 It is thus the objective of this study to search for common denominators and gain valuable insights for the benefit of different stakeholders, first and foremost developers, product managers and lawyers.

B. Qualitative Interviews and Research

I. Questions and Methodology

6 To accomplish this aim, a questionnaire was created and used as common thread during a series of interviews. Sixteen stakeholders[7] agreed to be interviewed, including (legal) representatives of projects, independent consultants, product managers, independent and employed FOSS developers and one professor of computer science with a special focus on open source software,[8] thus constituting a representative sample of interested stakeholders. In addition, further research was undertaken in the fields of law, organizational science and business informatics to back up the results obtained.

7 As expected, a clear distinction could be made between projects that actively managed contributions and those that did not. The following paragraphs shall provide an overview of selected FOSS projects, examining their makeup and reasons for using or refusing particular contributor agreements.

C. Projects That Do Not Actively Manage Contributions

I. The Linux Kernel

8 The Linux Kernel is only a small part of the software on a full Linux system, not including systems software, libraries or applications, but as the core, it is responsible for managing the hardware, running user programs and maintaining the overall security and integrity of the system.[9]

9 The Kernel was originally written by Linus Torvalds, who published it as a pet project on a usenet posting in August 1991.[10] At first it was released under its own licence, which had a restriction on commercial activity; however, Torvalds soon changed the license to the GPL 2, encouraging thousands of developers to actively contribute.[11] Today the Kernel is celebrated as the most important open source project in history, not only running on desktops, smartphones, routers, web servers, supercomputers, TVs, refrigerators, tablets and even the stock market (London, NY, Johannesburg, etc.), but in many areas being the undisputed leader.[12]

10 Legally, the Kernel may be regarded as a "composite work"[13] comprised of Linus Torvalds' original code with extensions and modifications contributed by other developers.[14] Torvalds thus holds the copyright

of the composite work.[15] As such, he can "distribute and reproduce"[16] the contributions as part of the composite work. However, he cannot modify or re-license the code under any license that goes against the rules of the individual works, nor can he defend possible violations in a court of law. To date every contributor is asked to provide his patches under the GPL v. 2, which is incompatible with many other outbound licenses including the GPL v. 3. Consequently, Torvalds or any other successor is not able to re-license – i.e. change – the outbound license or defend possible violations in a court of law unless he obtains the permission of all contributors in the form of contributor agreements.

11 But Torvalds vehemently refuses to adopt contributor agreements

> *not because they allow organizations to re-license, but because the copyright assignment paperwork ends up basically killing the community. Basically, with a CLA, you don't get the kind of "long tail" that the kernel has of random drive-by patches. And since that's how lots of people try the waters, any CLA at all – changing the license or not – is fundamentally broken.*[17]

12 Linus emphasizes that the Kernel benefits from many "drive-by-developers" who would be deterred if they were requested to sign contributor agreements before being able to submit a patch. It would further mean a disproportionate administrative outlay for Linus lieutenants. The "trusted lieutenants"[18] are roughly a dozen hackers responsible for maintaining a part of the Linux Kernel. Many developers send their patches directly to them instead of Linus.

II. Perl

13 The general-purpose Unix scripting language Perl is in a similar situation.

14 Perl was originally developed by Larry Wall in 1987 and published under the Artistic License v. 1,[19] an open source license, likewise developed by Larry Wall. Thousands of programmers used and improved Perl, turning it into one of the most widely known open source programming languages.[20]

15 The development process is overseen by Larry Wall and a small group of main developers called the "pumpkings". They make the day-to-day decisions on where Perl should go and make releases. Below that are the people with commit access to the repositories, who filter and apply patches and changes. Beyond that are the general community and contributors who submit patches and participate in the mailing lists. [21]

16 In the year 2000, Larry Wall and Alison Randall, decided to redesign Perl v. 5 and adapt it to the challenges of the 21st century. They attempted to migrate the project from the Artistic License v. 1 to v. 2, which was a legally overseen re-draft of version 1, and to convince every contributor to sign a CLA. However, instead of redesigning Perl 5, Perl 6 turned into a completely new language with a completely new developer community. This was largely due to the fact that most Perl 5 developers, estimated at around 500,000, refused to agree to a license change and boycotted the signing of CLAs.[22] As a result of the Perl language split, Perl 5 continues to be developed , now having arrived at v. 5.18, and Perl 6 has multiple implementation projects such as Rakudo Perl, which is based on Parrot and NQP (Not Quite Perl).[23] In order to be able to contribute to Perl 6, developers are required to sign a CLA, whereas developers of Perl 5 continue to follow the inbound=outbound approach.

III. LLVM

17 LLVM (formerly Low Level Virtual Machine) is a compiler infrastructure written in C++ designed for compile-time, link-time, run-time and "idle-time" optimization of programs written in arbitrary programming languages. Languages with compilers that use LLVM include ActionScript, Ada, D, Fortran, OpenGL Shading Language, Haskell, Java bytecode, Julia, Objective-C, Python, Ruby, Rust, Scala and C#.[24]

18 The LLVM project started in 2000 at the University of Illinois at Urbana–Champaign as a research infrastructure to investigate dynamic compilation techniques for static and dynamic programming languages. It was released under the University of Illinois/NCSA Open Source License, a non-copyleft license.[25]

19 The LLVM project managers decided against introducing contributor agreements and reasoned as follows:

> *The LLVM project does not require copyright assignments, which means that the copyright for the code in the project is held by its respective contributors who have each agreed to release their contributed code under the terms of the LLVM License.*
>
> *An implication of this is that the LLVM license is unlikely to ever change: changing it would require tracking down all the contributors to LLVM and getting them to agree that a license change is acceptable for their contribution. Since there are no plans to change the license, this is not a cause for concern.*
>
> *As a contributor to the project, this means that you (or your company) retain ownership of the code you contribute, that it cannot be used in a way that contradicts the license (which is a liberal BSD-style license), and that the license for your contributions won't change without your approval in the future.*[26]

IV. Outlook

20 The outbound = inbound approach is the very nucleus of open source programming. It was Stallmann's vision to free software development from appropriation through copyrights and patents.[27] To achieve this aim he developed a copyright license, the General Public License (GPL),which drew on the existing copyright regime to ensure exclusive rights for the public at large and not just for the original copyright holder. The only condition he imposed was that any derivative works and combinations of GPL licensed code should also be published under the GPL. For that reason the GPL has often been referred to as "viral".[28]

21 This virality helped volunteer communities come together on an informal basis to exchange ideas and build upon each other's work,[29] resting assured that this collective work and the license behind it would be enforced by the courts.[30]

22 The above projects were initiated during a *Zeitgeist* of free procreation of code – formalities such as contributor agreements were unknown. Companies were still sceptical, but allowed a couple of developers to write code in their working time if that saved money or raised efficiency. As a result, there is now an enormous user and developer base.

23 The management of these projects knows that they would be able to manage the projects much more efficiently by holding, or having particular usage rights, of the individual copyrights. However, this dilemma is accepted as given, since the administrative burden of introducing contributor agreements would hinder creativity and the acquisition of "eyeballs"[31] for effective bug detection.

24 This approach is thus perfectly acceptable for young and small projects wanting to test the ground and explore their creativity. Should the project take off, there is no obstacle to commercialization as such, if no contributor agreements have been requested – it all depends on the outbound license used.[32]

25 According to *Schaarschmidt et al.*,[33] the outbound = inbound approach is also suitable for R&D alliances not interested in paying expensive lawyers for drafting complicated contracts on the distribution of the intellectual property rights of the ensuing products.[34] Instead, everything is regulated by the public license. Thereby the completed product belongs to the community, and its source is open and visible for everybody. Depending on the nature of the public license, firms can practise open innovation protection to different degrees: should they use a strong copyleft outbound license, e.g. the AGPL or the GPL, they are no longer able to market their investment directly; however, the competition is also barred from doing so. Should they, on the other hand, use a permissive license, e.g. Apache or BSD, all parties can appropriate the code and include it into commercial products without having to share their changes with the public. [35]

26 Contributor agreements only become relevant when it comes to the management of the project, e.g. the ability to re-license the code under a different public license, to sublicense the code under a certain trademark or the ability to enforce possible violations in a court of law.

27 In theory, a project could also decide to introduce contributor agreements at a later stage. KDE, for instance, introduced its Fiduciary License Agreement (FLA) nearly ten years after its first release.[36] This is unproblematic where the number of committers is manageable. But for very big projects, it requires sure instincts to know when the crossroads is reached after which the perceived benefit of having contributor agreements is outweighed by the burden of seeking out untrackable developers. The latter, however, should apply to only a very small percentage of projects, given that less than 10% of all projects have more than 1,000 active committers at any given time. Most projects have only one to three committers.[37]

28 It may thus be concluded that certain projects, in this paper exemplified as Linux, Perl or LLVM, made a conscientious choice of not introducing contributor agreements in order to save on administrative resources and open the door for a flourishing community of developers. Due to their tremendous size, however, a change in the managerial approach is no longer conceivable. Smaller projects, by contrast, always have the choice of starting out without contributor agreements and introducing them at a later stage, should this be desirable.

D. Projects That Actively Manage the Intellectual Property of Contributions Through Contributor Agreements

29 For other projects, legal certainty, ability to enforce or flexibility to use the code outweigh the outbound licensing terms outweigh the administrative burden.

30 Those projects are governed by

1. foundations,
2. development partnerships (co-operatives) and
3. individual companies (single-vendor projects).

31 A selection of those projects shall be presented below.

I. Foundations

1. The FSF

32 Although Stallmann may be regarded as the forefather of the "inbound= outbound" approach, he soon abandoned this path for his own projects. He believed that the ability to re-license the code and enforce the GPL terms in a court of law[38] were fundamental to ensure a defensive free software regime.

33 To that end he created the Free Software Foundation (FSF), a neutral organization entrusted with the administration and enforcement of the copyrights in the ensuing collaborative works. In order for the FSF to become copyright holder of these works, each contributor is asked to sign a Contributor Assignment Agreement (CAA) transferring his ownership rights in the respective contribution to the foundation. For some GNU packages,[39] the FSF does not accept contributions of developers who have not signed a CAA. Problematic in this respect is that some jurisdictions do not accept outright transfers of ownership in copyright,[40] rendering the CAA in those jurisdictions most probably unenforceable. [41]

34 This problem is most salient in Europe, where the FSFE,[42] a sister organization of the FSF, provides legal support for developers and project managers.

KDE e.V.

35 A prominent protégé of the FSFE is KDE, e.V., whose community builds the graphic user interface (desktop) for Linux- or Unix-based operating systems.[43]

36 KDE is the prototype of a community-initiated project. Ever since the project started, the community has been driven by the creativity of the volunteers who contribute to the project. The administrative affairs of KDE are governed by the board, but there is no steering or central control for the development direction. The freedom of the code and independence of the developers is paramount.[44]

37 KDE licenses the ensuing code under the LGPL for the core framework and the GPL for applications ensuring that the code remains open for the community and is not appropriated by a third party.[45] Although the software produced in this way is not marketable as such, many businesses provide support, services and training around the freely downloadable software. Famous examples constitute the distributors Mint, Kubuntu and Debian.[46]

38 In line with the FSF(E)'s ideals, KDE e.V. takes up the role of fiduciary for its developers and asks, but does not compel, everybody to sign a Fiduciary License Agreement (FLA). This agreement is *strictu sensu* a CAA, since it triggers the transfer of ownership of the contribution to KDE. But it also has a fall-back clause: should ownership in the copyright not be transferable due to compulsory national laws, an exclusive license is granted.[47]

> *§ 1 Grant*
>
> *[..]Beneficiary assigns to KDE e.V. the Copyright in computer programs and other copyrightable material world-wide, or in countries where such an assignment is not possible,*
>
> *grants an exclusive licence, including, inter alia:*
>
> *1. the right to reproduce in original or modified form;*
>
> *2. the right to redistribute in original or modified form;*
>
> *3. the right of making available in data networks, in particular via the Internet, as well as by providing downloads, in original or modified form;*
>
> *4. the right to authorize third parties to make derivative works of the Software, or to work on and commit changes or perform this conduct themselves.*

39 As a fiduciary, KDE is interested in sustaining the project and ensuring its longevity. Accordingly, two main tools are necessary to achieve this aim: the ability to 1) re-license[48] and adapt the project to new technological circumstances and 2) defend the project and its developers in its own name:[49]

> *§ 3 K DE e.V.'s Rights and Re-Transfer of Non-Exclusive Licence*
>
> *KDE e.V. shall exercise the granted rights and licences in its own name. Furthermore, KDE e.V. shall be authorized to enjoin third parties from using the software and forbid any unlawful or copyright infringing use of the Software, and shall be entitled to enforce all its rights in its own name in and out of court. KDE e.V. shall also be authorized to permit third parties to exercise KDE e.V.'s rights in and out of court.*

40 KDE, in line with the FSFE, chose CAAs, or exclusive licenses, because it believes that simple, non-exclusive CLAs are not as effective when going to court or trying to re-license.[50]

41 The former has recently been confirmed by *Engelhardt*.[51] The latter, however, is being circumvented expressly and impliedly by other projects discussed below, an indication that in the absence of common standards and/or judicial precedents, legal uncertainty as to the effects of CAAs and CLAs is still common.

2. The Open Source Initiative (OSI)

42 With the birth of the OSI[52] and the proliferation of public licenses, the open source business model grew popular with companies that had previously been sceptical and hostile due to the viral effect of the free software. Permissive licenses, such as the Apache or the BSD license, however, encouraged companies interested in displacing established

software companies to form alliances or sponsor open source projects.[53]

43 As some of these companies were fierce competitors,[54] the idea to outsource the administrative affairs and intellectual property issues for the ensuing product to a neutral, non-profit organization began to gain momentum.

a.) Apache

44 One of those organizations is Apache, a US 501(c)(3) non-profit corporation which provides organizational, legal and financial support for a broad range of over 140 open source software projects.[55]

45 Projects that have been admitted as Apache projects are promoted under the Apache license, a permissive license that allows companies to take the open source infrastructure, change it and subsume it into closed source projects. The Apache license, for instance, would be recommendable for the development of a reference implementation for a standard.[56] Thereby the competition is shifted from the infrastructure market to the market for applications and complementary products.[57]

46 Companies or individual developers engage in particular Apache projects because they are interested in supporting the quality of the Apache trademark. It allows them to vouch for the openness and quality of the software they use within their end-products.[58] Some, in turn, contribute for intrinsic reasons, wanting to give something back to the community.

47 Since Apache caters for many commercially oriented companies that form R&D alliances under its auspices, it has a strong interest in being able market the code under its trademark and vouch for the provenance of the code. As such, a prerequisite of becoming a committer and being able to submit patches is to sign an Individual[59] Contributor License Agreement(ICLA).[60]

48 Apache rejects CAAs as these are difficult to obtain.[61] The companies for which most individual developers work do not want to part with the intellectual property of the individual contributions in case they are patentable or otherwise commercially applicable.[62]

49 Through the CLA, however, they retain the intellectual property rights, and grant Apache:[63]

> *a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, sublicense, and distribute [their] Contributions and such derivative works.*
>
> *There is, however, no explicit right to enforce the copyright in a court of law. This may be explained with the fact that Apache, and projects that follow Apache's example, are not too keen to be involved in copyright infringement claims.[64] Since the code is designed to be used within proprietary products, all that is required is a copyright notice and the preparedness to provide the source code upon request. Hitherto cease and desist letters were sufficient to secure this outcome.*
>
> *Since non-exclusive licenses do not automatically confer standing in a court of law, however, it might be recommendable to include such a right expressly in a CLA, simply to be in the position to sue should it become necessary at some point.*

II. Development Partnerships (Co-operatives)

50 More and more (commercial) software customers are dissatisfied with what they perceive as "vendor lock-in" and join forces to commission open source software solutions that replace individually grown strategic IT systems and are flexible enough to meet the challenges of the future.

51 These alliances are generally organized as development partnerships or co-operatives in a specific economic area.

52 Particularly problematic is e.g. the maintenance and development of energy and water networks[65] in light of the transition from the fossil fuel and nuclear energy age to the solar and efficient energy age (*Energiewende*). This is largely due to the fact that the IT-systems landscape dates back to a time when software developers designed monolithic, proprietary systems that were unable to interact with each other or allow for new functionalities without exposing the providers to considerable expenses.

53 A solution is the commissioning of open source solutions which use service-oriented architecture (SOA) to break the monolithic software incrementally and integrate it as separate components.

54 Other fields include e.g. the automotive industry in the form of AUTOSAR,[66] which is an open and standardized automotive software architecture jointly developed by automobile manufacturers, suppliers and tool developers whose objective is to create and establish open standards for automotive E/E (Electrics/Electronics) architectures.[67]

55 Such user co-operatives place orders with different IT providers who develop new functionalities and/or cross-system interfaces. In order to avoid future vendor lock-in, a prerequisite is the transfer of copyrights in the developed work to the user co-operative in the form of a CAA, akin to a "quasi-

employment relationship". Since research in this area is still in its infancy, and details as to the exact wording of the CAA are being held confidential, no further information could be retrieved.[68]

III. Individual Companies (Single Vendor Open Source Software Project)

56 Single-vendor commercial open source software projects are projects that are owned by a single firm that derives a direct and significant revenue stream from the software. Using a single-vendor open source approach, firms can get to market faster with a superior product at lower cost than possible for traditional competitors.[69]

57 Where a firm decides to open-source previously closed source software (firm-initiated – single vendor project),[70] it will most certainly want to be able to have the intellectual property rights in the contributions in order to effect different commercialization strategies and business models.[71]

58 Single vendors thus tend to use GPL licenses and request extensive contributor agreements in the form of CAAs or CLAs that allow them to pursue a dual licensing strategy.[72] This approach is particularly smart since code developed under the GPL does not normally lend itself to being commercialized in different ways. The traditional business model around GPL licensed code is the provision of support, services and training, as offered by the Linux distributor Red Hat. Due to the viral nature of the GPL, it is impossible to include GPL-licensed code in proprietary products. The dual licensing strategy, however, opens new revenue streams for the initiating firm by e.g.

59 including the code in proprietary products of their own and selling commercial licenses to competitors.

1. CAA or CLA?

60 A widespread belief is that in order to be able to sublicense GPL-licensed code under any license, including commercial licenses, there would have to be an outright transfer of ownership in the form of a CAA.

61 Many projects therefore choose an outright transfer of ownership – including the fall-back option of granting an exclusive license should local copyright laws not allow a straight transfer of ownership.[73] Others, however, bend the wording of a CLA to the extent they reserve the rights normally only attainable under a CAA. An example of the former approach is ETAS,[74] a company that provides engineering services, consulting, training and support for the development of embedded systems for the automotive industry; an example for the latter is Digia, the owner of the programming environment QT, which will be explored later.

2. ETAS

62 ETAS and Robert Bosch Engineering and Business Solutions (RBEI) jointly published BUSMASTER,[75] a free open source PC software that allows for flexible modification and extensions regarding bus systems, protocols and hardware interfaces. The current BUSMASTER version is based on the preceding software tool CANvas, conceptualized, designed and developed by RBEI.

63 When the company decided to open source the code, they chose the LGPL, which permits the provision of proprietary add-ons that can be dynamically linked to the open source core.

64 In addition, they opted for a CAA based on Harmony v. 1.0.[76] The main reason for choosing the CAA was to be able to adapt the project to new circumstances, e.g. if at some point it might be beneficial for the project to be turned into an Eclipse project, there would have to be a licensing change from the LGPL to the Eclipse Public License (EPL), which are incompatible and could not be effected without the permission of all contributors.[77]

65 Of course, owning the copyright in the contributions ETAS is automatically able to use the code in proprietary products, to defend violations in a court of law and to license the code commercially to third parties without having to explicitly state it in their CAA, although they do so:[78]

> *We may license the Contribution under any license, including copyleft, permissive, commercial, or proprietary licenses...*

66 The outright transfer of ownership has, however, often been criticized as too restrictive for two main reasons:[79] It bars developers (and their companies) from exploiting their contributions otherwise, e.g. by contributing to a different project, using it in a commercial distribution or applying for a patent. In addition, there is the constant danger of the project management changing its business strategy and converting the open source project into a commercial one.

67 In the following paragraphs these points shall be discussed, highlighting the solutions hitherto developed under Harmony v. 1[80] and as such adopted by ETAS, or where appropriate by contributoragreements.org.[81]

a.) The Contribution Cannot Be Exploited Otherwise

68 Some critics of CAAs argue that by requesting a CAA, the original developer is barred from exploiting the contribution otherwise. Particularly Apache, referred to above, stated that they chose CLAs since CAAS were too difficult to obtain. However, this point of criticism could be mitigated by providing a generous license back to the contributor. For instance, this could take the form envisaged by ContributorAgreements.org,[82] a project dedicated at the standardization of contributor agreements:

> *Upon such grant of rights to Us, We immediately grant to You a worldwide, royalty-free, non-exclusive, perpetual and irrevocable license, with the right to grant or transfer an unlimited number of non-exclusive licenses or sublicenses to third parties, under the Copyright covering the Contribution to use the Contribution by all means, including, but not limited to:*
>
> *to publish the Contribution,*
>
> *to modify the Contribution, to prepare Derivative Works based upon or containing the*
>
> *Contribution and to combine the Contribution with other software code,*
>
> *to reproduce the Contribution in original or modified form,*
>
> *to distribute, to make the Contribution available to the public, display and publicly perform the Contribution in original or modified form.*[83]
>
> *This license back is limited to the Contribution and does not provide any rights to the Material.*

69 Given this wording, the developer has *prima facie* all the rights he would have had he only ever signed a CLA.

70 Another way forward could be a joint, independent copyright assignment. This approach allows each individual party to use the contribution as the "quasi-owner", in the words of one interviewee:

> *one party can do whatever they want with licensing in the future and the other party can do whatever they want – it's like having two separate works.*[84]

71 A famous example using joint, independent copyright assignments was Sun/Oracle:

> *Contributor hereby assigns to Sun joint ownership in all worldwide common law and statutory rights associated with the copyrights, copyrights application, copyright registration and moral rights in the contribution to the extent allowable under applicable local laws and copyright conventions. Contributor agrees that this assignment may be submitted by Sun to register a copyright in the contribution. Contributor retains the right to use the contribution for Contributor's own purposes.[..]*[85]

b.) Project Management Might Close the Open Source Project

72 Unfortunately, Oracle changed its business strategy after acquiring SUN and "closed" open Solaris, an open source operating system with the ability to become a serious competitor to Linux. This left thousands of developers owning a part to an unattainable whole and evoked the anger of the community.[86]

73 Drafters of CAAs have thus suggested ensuring that the transfer of ownership takes place only upon the condition that the project will always maintain an open source branch. For instance, this could be framed as follows:

> *As a condition on the exercise of this right [to use the contribution under any license], We agree to also license the Contribution under the terms of the license or licenses which We are using for the Material on the Submission Date*[87]

74 This approach was suggested by Harmony's CAA v. 1.0 and is currently used by ETAS.

75 ETAS thus reserves the right to use the contribution under any license; however, it grants a broad license back to the developer and ensures there will always be a branch under the LGPL, the open source license in force on the submission date.

3. Digia

76 Another approach might be the use of a CLA reserving far-reaching rights. A prominent example is Digia,[88] the owner of the programming environment QT.[89] Digia is a Finnish company which not only provides commercial support, services and training around QT, but also distributes the GPL-licensed code under commercial licenses.[90] These allow interested parties to modify and extend the code without having to make the changes available to the public. Digia requires every developer to sign a CLA in which he agrees to license his contribution and give Digia a

> *sublicensable, irrevocable, perpetual, worldwide, non-exclusive, royalty-free and fully paid up copyright and trade secret license to reproduce, adapt, translate, modify, and prepare derivative works of, publicly display, publicly perform, sublicense, make available and distribute (the) Licensor's Contribution(s) and any derivative works thereof under license terms of Digia's choosing including any Open Source Software license.*

77 Digia is thus granted a non-exclusive license which conveys the right to sublicense and make available the code under any license of Digia's choosing, i.e. a right to re-license may also be inferred.

78 Digia is further aware that a multi-licensing business model is not feasible without being in the position to

pursue enforcement of the code in front of a court of law:

> *3.5 Enforcement Authorization*
>
> *The Licensor hereby authorizes, and agrees to execute without undue delay any and all documents reasonably necessary to effect such authorization, for Digia to enforce the Licensor's copyrights in and to a Licensor Contribution on the Licensor's behalf against any third parties as Digia at its discretion deems appropriate, at Digia's expense.*
>
> *In jurisdictions where such authorization is not possible under mandatory applicable law, the Licensor hereby undertakes upon Digia's request and at Digia's expense, to act jointly with Digia (as a co-plaintiff) in enforcing the Licensor's copyrights,[...]*[91]

79 In line with *Engelhardt's* assumptions that a non-exclusive license does not per se confer standing in a court of law, Digia expressly reserves the rights to have standing to defend possible violations in a court of law. Had Digia chosen a CAA, it would not have had to make these rights explicit.

80 Digia thus assumes it can obtain the same rights conveyed by an outright transfer of ownership through a CLA if they are expressly listed therein. Whether this is truly the case has not yet been tested.

81 It is also important to note that critics condemn far-reaching CLAs to the same extent as CAAs elaborated above. Although a CLA allows a contributor to otherwise exploit the contribution, there is always the danger of a single vendor abandoning the open source project and leaving a developer with a part to an unattainable whole. For CLAs it is therefore equally important to include a clause stating that any license grant takes place upon the condition that the project will always remain under a free and open source license.

IV. Outlook

82 From the aforesaid, one may conclude that there is a variety of foundations/cooperatives and single vendor open source businesses with contrasting ethos using different CAAs and CLAs for differing purposes.

83 Upon a closer look, however, it becomes clear that despite having different agendas, parties of contributor agreements generally have the same aim: owners want to be able to perform all acts exclusively reserved for copyright owners under copyright law, i.e. copy, distribute, modify and communicate to the public, but most importantly they want to be able to re- and sub-license the code to third parties, in some cases even under a different outbound license, and to defend possible violations in a court of law.

84 Developers in turn want to retain the right to use the contribution in another project, possibly even in a commercial application or even a patent, and be sure that the open source project will always remain under a free and open source license and not become a victim of a business strategy change.

85 Since there is no accepted standard definition of what a contributor agreement should contain in order to have a particular effect, legal departments constantly re-invent the wheel and draft contributor agreements either based on outdated assumptions or adventurous developments of the law. These reduce the understandability and add to the confusion and distrust of developers and their respective employers.

86 It might thus be time to start thinking about a standardization effort by means of an open source contributor agreement platform, where interested parties come together and decide what infrastructure should underlie every contributor agreement. These parties should include developers, project managers, product managers and lawyers.

87 It should have a modular architecture, so that interested parties could add individual conditions and rights depending on their particular needs. All of these modules would be endorsed by a legally qualified committee, thus ensuring that the use of a contributor agreement of said format would be a qualitatively high legal document produced in the transparency of the open source process.

E. Conclusion

88 To conclude, it is safe to say that the divide between projects which use outbound as inbound and those which actively manage intellectual property rights is (currently) here to stay.

89 It would be tilting against windmills to try to convince the unconviceable of using contributor agreements of any sort. Neither the standardization nor possible automatization of rights management is in these projects' interest as it would mean an increased administrative burden, i.e. costs, which would be difficult to raise.[92]

90 On the other hand, more and more projects have an interest in being able to actively manage the intellectual property of their contributions. Due to the lack of a common standard, however, legal departments constantly re-invent the wheel, resulting in a very unhomogenous contributor agreement landscape, prone to distrust and criticism.

91 The current paper thus proposes a standardization effort, using the very same open source method to

create an acceptable infrastructure for understandable and effective contributor agreements.

1 Perl Foundation, http://www.perlfoundation.org/, Linux Foundation, http://www.linuxfoundation.org/, Mozilla Foundation, https://www.mozilla.org/en-US/foundation/, accessed 16 May 2014.

2 Busmaster of ETAS, http://www.etas.com/de/products/applications_busmaster.php, accessed 16 May 2014.

3 Maracke, C., Editorial: Copyright Management for open collaborative projects: Inbound Licensing Models for open Innovation, Volume 10, Issue 2, August 2013, p. 143, http://script-ed.org/wp-content/uploads/2013/08/editorial.pdf, accessed 7 May 2014.

4 Maracke, C., Metzger, A., Concept Paper: Network of Stewards for Free and Open Source Software projects, p. 2, http://contributoragreements.org/wp-content/uploads/2013/05/ConceptPaper_NetworkV9_Abstract.pdf, accessed 16 May 2014.

5 Famous examples constitute e.g. the Linux Kernel, Perl 5 or the LLVM Project .

6 Much research on the topic is conducted in the field of computer science and economics; see e.g. Riehle, D. Three Positions on the Future of Open Source Research, p. 2, http://dirkriehle.com/wp-content/uploads/2010/01/FOSS-2010-Position-Paper.pdf, Controlling and Steering Open Source Projects, IEEE Computer Society, July 2011, p. 94.

7 A full transcript of any given interview may be available upon request: sylviafjakob@live.de.

8 Riehle, D., Friedrich – Alexander University of Erlangen (Nürnberg, Germany).

9 Corbeth, J., Kroah-Hartman, G. & McPherson, A., Linux Kernel Development: How Fast It Is Going, Who Is Doing It, What They Are Doing, and Who Is Sponsoring It, A white paper by the Linux Foundation, 2009, http://www.linuxfoundation.org/sites/main/files/publications/whowriteslinux.pdf, accessed 14 May 2014.

10 https://groups.google.com/forum/#!msg/comp.os.minix/dlNtH7RRrGA/SwRavCzVE7gJ, accessed 4 June 2014.

11 http://web.archive.org/web/20110721105526/http://www.kernel.org/pub/linux/kernel/Historic/old-versions/RELNOTES-0.12, accessed 4 June 2014.

12 Contreras, F., Why the Linux Kernel is the most important project in history, http://felipec.wordpress.com/2011/03/06/why-linux-is-the-most-important-software-project-in-history/, accessed 16 April 2014.

13 17 U.S.Code § 201 c).

14 Linus as copyrightholder of the composite Linux Kernel: http://yarchive.net/comp/linux/collective_work_copyright.html, accessed 20 May 2014.

15 Discussion of Contributor Agreements between Linus Torvalds, Greg Kroah-Hartmann & others on Google + on 20 January 2014, https://plus.google.com/111049168280159033135/posts/NstZfwXbAti, accessed 2 April 2014.

16 17 U.S.Code § 201 c) other jurisdictions?

17 17Discussion of Contributor Agreements between Linus Torvalds, Greg Kroah-Hartmann & others on Google + on 20 January 2014, https://plus.google.com/u/0/111049168280159033135/posts/NstZfwXbAti, accessed 2 April 2014.

18 Dafermos, G., On the fourfold structure, http://p2pfoundation.net/Linux_-_Governance, accessed 16th April 2014

19 http://opensource.org/licenses/Artistic-1.0, accessed 16 April 2014; see further Jacobsen v. Katzer for the enforceability of the Artistic License, http://www.cafc.uscourts.gov/images/stories/opinions-orders/08-1001.pdf, accessed 16 April 2014.

20 Large projects written in Perl include cPanel, Slash, Bugzilla, RT, TWiki, and Movable Type; high-traffic websites that use Perl extensively include bbc.co.uk, Priceline.com, Craigslist, IMDb, LiveJournal, DuckDuckGo,Slashdot and Ticketmaster. It is also an optional component of the popular LAMP technology stack for web development, in lieu of PHP or Python; see http://en.wikipedia.org/wiki/Perl, accessed 16 April 2014.

21 http://www.perlfoundation.org/how_tpf_works, accessed 16 March 2014.

22 Perl Foundation President, personal communication, August 2013.

23 http://en.wikipedia.org/wiki/Perl_6.

24 http://en.wikipedia.org/wiki/LLVM.

25 http://llvm.org/.

26 http://llvm.org/docs/DeveloperPolicy.html, accessed 16 April 2014.

27 Jaeger/Metzger, Open Source Software, Rechtliche Rahmenbedingungen der Freien Software, 3. Auflage, Verlag C.H. Beck, 2011, p.12.

28 See e.g. the court in *Versata v. Ameriprise, p. 9* http://de.scribd.com/doc/212507936/Versata-Software-v-Ameriprise, accessed 16 May 2014.

29 See e.g. Lerner & Tirole, Some Simple Economics of Open Source,THE JOURNAL OF INDUSTRIAL ECONOMICS 0022-1821,Volume L, June 2002 No., p. 201,202, http://onlinelibrary.wiley.com/doi/10.1111/1467-6451.00174/pdf, accessed 15 June 2013, Gonzales- Barahona, J.M., Robles, G., Trends in Free, Libre, Open Source Communities: From Volunteers to Companies, IT 55 (2013) 5, Oldenbourg Wissenschaftsverlag, p. 173- 180, p. 174.

30 See e.g. http://www.pro-linux.de/news/1/11486/gericht-skype-verletzt-die-gpl.html, accessed 16 May 2014.

31 Famous quote of Raymond, E.S.: "Given enough eyeballs, all bugs are shallow" (Linus' Law), in The Cathedral and the Bazaar, 1999, p. 22

32 Where a strict copyleft license is used, for instance the AGPL, the code cannot be incorporated into closed source products, but many other business models can be pursued. Where a weak copyleft outbound license has been used, for instance the BSD license, any sort of commercialization is allowed.

33 Schaarschmidt, M., Bertram, M., Zerwas, D.& Kortzfleisch, H., Kommerzialisierungsansätze in Open Source Software Projekten, HMD, 283, p. 6-16,p. 12, http://download.springer.com/static/pdf/209/art%253A10.1007%252FBF03340658.

34 See e.g. AMQP.org, an alliance of blue chip companies that developed an advanced message queuing protocol, which was recognised as an international standard and does not request CAs of any sort: http://www.amqp.org/, accessed 16 May 2014.

35 See e.g. the Apache License v. 2, http://oss-watch.ac.uk/resources/apache2 or the BSD license http://oss-watch.ac.uk/resources/modbsd, accessed 15 May 2014.

36 "At the General Assembly of KDE e.V. in August 2008 the membership voted to adopt a Fiduciary Licensing Agreement as the preferred form for assigning copyright to KDE e.V.: http://ev.kde.org/rules/fla.php, accessed 24 June 2014.

37 http://bit.ly/1Bk5Vld

38 Metzger, A., Internationalisation of FOSS contributory Copyright Assignments and Licenses: Jurisdiction-Specific or "Unported", SCRIPTed, Vol. 10, Issue 2, August 2013, p. 178-206, p. 179.

39 Metzger, A., Internationalisation of FOSS Contributory Copyright Assignments and Licenses: Jurisdiction-Specific or "Unported", SCRIPTed, Vol. 10, Issue 2, August 2013, p. 178-206, p. 179.

40 See e.g. § 29 German Copyright Law.

41 https://fsfe.org/work.en.html, accessed 14 May 2014.

42 https://fsfe.org/index.de.html, accessed 14 May 2014.

43 http://www.kde.org/, accessed 15 April 2014.

44 Former Board Member of KDE, Mirko Boehm,personal communications, September 2013.

45 Ibid.

46 http://www.kde.org/download/distributions.php, accessed 12 May 2014.

47 http://ev.kde.org/resources/FLA-prefab.pdf, accessed 24 February 2014.

48 http://ev.kde.org/resources/FRP.pdf, accessed 24 February 2014.

49 http://ev.kde.org/resources/FLA-prefab.pdf, accessed 24 February 2014.

50 Legal Counsel, FSFE, personal communication, October 2013.

51 Engelhardt, T., Drafting Options for Contributor Agreements for Free and Open Source Software: Assignment, (Non) Exclusive Licence and Legal Consequences. A Comparative Analysis of German and US Law, SCRIPTed, Volume 10, Issue 2, August 2013, p. 148-176, p. 164-65.

52 http://opensource.org/, accessed 14 March 2014.

53 Riehle, D., The economic case for Open Source Foundations, IEEE, January 2010, p. 86-90, p. 88.

54 Barahona-Gonzales, J., Trends in Free, Libre, Open Source Software Communities: From Volunteers to Companies, IT, 55 (2013) 13, p. 173-180, p. 175, Oldenbourg Wissenschaftsverlag.

55 http://projects.apache.org/, accessed 27 January 2014.

56 The standardization of the Java Portlet Specification 3.0 under the sponsorship of IBM is being developed, at least in parts, under the auspices of the Apache Foundation.

57 Personal communications with Java Portlet Specification Manager, Martin Scott Nicklous, August 2013.

58 Riehle, D., Controlling and steering open source projects, IEEE Computer Society, July 2011, p. 96.

59 https://www.apache.org/licenses/icla.txt, accessed 7 March 2014.

60 For those who are employed by a particular company, the employer has to sign a Corporate CLA (CCLA): http://www.apache.org/licenses/cla-corporate.txt, accessed 7 March 2014.

61 Personal communication with board member of the Apache Foundation, Bertrand Delacretaz, August 2013.

62 Ibid.

63 https://www.apache.org/licenses/icla.txt, accessed 24 February 2014.

64 Personal communication with Apache board member Bertrand Delacretaz, August 2013, and Django Foundation President, Russell Keith- Maguee, October 2013.

65 Herdt, P., Konsortiale Software Entwicklung im Energiesektor, http://www.osbf.eu/blog/konsortiale-open-source-softwareentwicklung/im-energiesektor/#.U3ZGG1d-r6M, 26 November 2013, accessed 15 March 2014.

66 AUTomotive Open System Architecture, http://www.autosar.org/, accessed 15 April 2014.

67 http://en.wikipedia.org/wiki/AUTOSAR, accessed 16 May 2014.

68 Herdt, P., supra, p. 6

69 Riehle, D., The commercial open source business model, http://dirkriehle.com/publications/2009-2/the-commercial-open-source-business-model/, accessed 18 November 2013.

70 Riehle, D., The single vendor Commercial Open Source Business Model, Information Systems and e-Business Management archive, Volume 10 Issue 1, March 2012, p. 5-17, p. 5.

71 Schaarschmidt et al., supra, p. 11.

72 See also Comino,S. & Manenti, F.M., Dual licensing in Open Source Software markets, http://www.webmeets.com/files/papers/EARIE/2009/244/CominoManenti_Dual_licensing.pdf, accessed 10 May 2014.

73 See KDE supra.

74 ETAS, http://www.etas.com/en/index.php?langS=true&, accessed 16 May 2014.

75 BUSMASTER, http://www.etas.com/en/products/applications_busmaster.php?langS=true&, accessed 16 May 2014.

76 CAA, https://raw.githubusercontent.com/rbei-etas/busmaster-documents/master/contributor_agreement_entity.pdf, accessed 24 November 2013.

77 Personal communication with the product manager of BUSMASTER, Dr. Tobias Lorenz, August 2013.

78 CAA, see note 73.

79 Personal communications with several stakeholders, August – October 2013.

80 http://harmonyagreements.org/, accessed 14 February 2014.

81 http://contributoragreements.org/, accessed 16 May 2014.

82 Ibid.

83 http://development.contributoragreements.org/, accessed 16 May 2014.

84 Personal communication with economist/ IT consultant, Joseph Potvin, Canada, August 2013.

85 https://www.openoffice.org/licenses/jca.pdf, accessed 16 May 2014.

86 Personal communication with IT student, Jens Kadenbach, August 2013, backed up by http://www.cnet.com/news/oracle-apparently-shuts-doors-on-opensolaris/, accessed 20 October 2013.

87 http://harmonyagreements.org/, accessed 16 May 2014.

88 http://qt-project.org/, accessed 16 May 2014.

89 http://www.digia.com/, accessed 16 May 2014.

90 http://www.digia.com/en/Home/Company/Press/2012/Digia-to-acquire-Qt-from-Nokia/, accessed 16 May 2014.

91 http://qt-project.org/legal/QtContributionLicenseAgreement.pdf. An interesting twist in Digia's licensing terms is the granting of "consideration", which turns the CLA prima facie into a sale; see ECJ in Usedsoft v. Oracle, C 128/11).

92 Just think of the 500,000 Perl 5 developers.