Enforcing open source software licenses as a contributor

Ilari Lahtinen
Master Thesis
University of Turku
Faculty of Laws
August 2020

UNIVERSITY OF TURKU
Faculty of Laws

Ilari Lahtinen: Enforcing open source software licenses as a contributor

Master Thesis, 70 p.
Intellectual property right law
August 2020
The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin OriginalityCheck service.

The importance of open-source software licenses has increased over recent years. One reason for the trend is the companies are more and more interested in open-source software. At the same time, with the increased use of the open-source license also disputes relating to them have increased. In my master thesis, I aim to discover who and within which limitations can raise a claim in the case of open-source software license infringement.

Main research questions in my thesis are what are requirements for the open-source software project contributor to raise a claim in case of license infringement and on the other hand in which extend the contributor can raise a claim. The thesis also discovers whether every contributor has to be involved in raising the claim and running the case in the court. In other words, the question is whether one contributor may run the case on behalf of all contributors as a class action or some else way. Another perspective in the thesis is in which form the contributor have copyrights to the open-source code they have contributed. It is discovered whether source code is considered as collective work, jointly authored work or adaptive work when contributions are made over time.

The thesis covers both relevant Finnish and European Union legislation. Also, some US legislation and case law are included as a preference because contributing open-source projects is an international phenomenon. The thesis also discovers what are the differences between legal and technical perspective when it comes to the open-source software development process and its result, the software.

The outcome of the thesis is that every contributor has to take part in the infringement dispute process because it is not possible to run class action about an intellectual property dispute. There is also no way to move right to raise a claim to someone other with the contract, and therefore, that kind of action is not possible in license infringement cases either. One point which is also noticeable is that to get the right to raise a claim; the contributor must contribute to the project such that the contribution is intellectual enough to be alone protected with the copyright.

Keywords: Open source license, Copyright, Intellectual property right litigation

TURUN YLIOPISTO
Oikeustieteellinen tiedekunta

ILARI LAHTINEN: Enforcing open source software licenses as a contributor

Pro Gradu -tutkielma, 70 s.
Immateriaalioikeus
Elokuu 2020
Turun yliopiston laatujärjestelmän mukaisesti tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck-järjestelmällä.

Avoimen lähdekoodin lisenssien merkitys ohjelmistotuotannossa on kasvanut viime vuosina. Syynä tähän on ollut yritysten kasvanut kiinnostus avointa lähdekoodia kohtaan. Samalla kun avoimen lähdekoodin ohjelmistojen käyttö on lisääntynyt myös niihin liittyvät lisenssiriidat ovat lisääntyneet. Tutkielmassa tarkoituksena on selvittää kuka ja missä laajuudessa voi nostaa kanteen lisenssirikkomuksen sattuessa.

Tutkielman keskeisimpinä tutkimuskysymyksiä käsittelevät sitä, miten avoimen lähdekoodin sovelluksen kehittämiseen osallistunut henkilö voi lähteä ajamaan kannetta lisenssin loukkaustilanteessa ja toisaalta mitkä ovat edellytykset tälläisen kanteen nostamiselle. Selvitettävänä on täytyykö kaikkien kehittäjien osallistua kanteen nostamiseen ja sitä mahdollisesti seuraavaan oikeusprosessiin vai voiko yksi kehittäjistä ajaa asiaa kaikkien puolesta esimerkiksi joukkokanteena. Tutkielmassa käsitellään myös sitä, millaisen kokonaisuuden avoimen lähdekoodin ohjelmisto muodostaa tekijänoikeusnäkökulmasta. Tässä erityisenä tutkimuksen kohteena on se, muodostuuko ohjelmistosta yhteisteos, kokoelmateos vai muunnettu teos, kun sen kehittämiseen osallistuu useita kehittäjiä mahdollisesti eriaikoina.

Tutkielma keskittyy Suomen ja Euroopan Unionin lain säädäntöön, mutta eräissä kohdissa myös Yhdysvaltojen lainsääntöä on käytetty vertailukohtana sillä avoimen lähdekoodin sovellusprojektit ovat yleensä kansainvälisesti kehitettyjä. Tutkielmassa tutkitaan myös, miten oikeustieteellinen näkökulma avoimen lähdekoodin sovelluksen syntymisestä poikkeaa teknisestä näkökulmasta ohjelmiston synnystä.

Tutkielma päätyy johtopäätökseen, että jokaisen avoimen lähdekoodin kehittäjän tulee omalta osaltaan ajaa kannetta lisenssiehtojen rikkomistilanteessa, koska suomalainen lainsäädäntö estää joukkokanteen nostamisen muissa kuin kuluttajansuojariidoissa ja lisäksi myöskään kanneoikeuden siirtäminen sopimuksen avulla ei ole mahdollista. Kanneoikeutta rajoittaa myös se, että kehittäjän panoksen ohjelmistoon täytyy itsessään olla sellainen, että se ylittää ohjelmiston lähdekoodille asetetun teoskynnyksen.

Asiasanat: Avoimen lähdekoodin lisenssit, Tekijänoikeudet, Immateriaalioikeusriidat

Contents

Official sources and bibliography

Articles

Alexander Hars SO, “Working for free? Motivations for participating in open-source projects” (2002) 6(3) International Journal of Electronic Commerce 25.

Crosby M and others, “Blockchain technology: Beyond bitcoin” (2016) 2(6-10) Applied Innovation 71.

Hamano JC, “GIT–A stupid content tracker” (2006) 1 Proc. Ottawa Linux Sympo 385.

Hemel A and Coughlan SM, “Making Sense Of Git In A Legal Context” (2017) 9 IFOSS L. Rev. 19.

Hertel G, Niedner S, and Herrmann S, “Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel” (2003) 32(7) Research policy 1159.

Husa J, “Valkoista yksisarvista pyydystämässä vai mörköä paossa – »oikeaa oikeusvertailua»?” (2010) 2010(5) Lakimies 700.

Lakhani KR and Von Hippel E, “How open source software works:“free” user-to-user assistance” in *Produktentwicklung mit virtuellen Communities* (Springer 2004).

Lerner J and Tirole J, “Some simple economics of open source” (2002) 50(2) The journal of industrial economics 197.

— “The scope of open source licensing” (2005) 21(1) Journal of Law, Economics, and Organization 20.

Madey G, Freeh V, and Tynan R, “The open source software development phenomenon: An analysis based on social network theory” [2002] AMCIS 2002 Proceedings 247.

Meeker HJ, “Open Source and the Age of Enforcement” (2012) 4 Hastings Sci. & Tech. LJ 267.

Mullins L, “Using metadata to support DRM, trading and administration of globally deployed digital products” (2009) 5(2) Journal of Digital Asset Management 75.

Nyman L and others, “Understanding Code Forking in Open Source Software: An examination of code forking, its effect on open source software, and how it is viewed and practiced by developers” [2015].

O’Neill JB and Gaspar CJ, “What Can Decisions by European Courts Teach Us About the Future of Open-Source Litigation in the United States” (2010) 38 AIPLA QJ 437.

O’Mahony S, “Guarding the commons: how community managed software projects protect their work” (2003) 32(7) Research policy 1179.

Oksanen V and Välimäki M, “Free software and copyright enforcement: A tool for global copyright policy?” (2006) 18(4) Knowledge, Technology & Policy 101.

Popek GJ and Goldberg RP, “Formal requirements for virtualizable third generation architectures” (1974) 17(7) Communications of the ACM 412.

Savelyev A, “Copyright in the blockchain era: Promises and challenges” (2018) 34(3) Computer law & security review 550.

Siau K and Tian Y, “Open Source Software Development Process Model: A Grounded Theory Approach” (2013) 21(4) Journal of Global Information Management (JGIM) 103.

Tichy WF, “RCS—a system for version control” (1985) 15(7) Software: Practice and Experience 637.

Udsen H, “Open source licences” in *User Generated Law* (Edward Elgar Publishing 2016).

Välimäki M, “Avoimen lähdekoodin ohjelmistolisensseistä” (2002) 5 Defensor Legis.

— “Introducing Class Actions in Finland: An Example of Law-making Without Economic Analysis”, in *The Law and Economics of Class Actions in Europe* (Edward Elgar Publishing 2012).

Welser M von, “Opposing the Monetization of Linux: McHardy v. Geniatech & Addressing Copyright Trolling in Germany” (2018) 10 IFOSS L. Rev. 9.

West J, “How open is open enough?: Melding proprietary and open source platform strategies” (2003) 32(7) Research policy 1259.

Books

Aarnio A, *Laintulkinnan teoria* (Werner Söderström Osakeyhtiö 1988).

— *Tulkinnan taito* (Werner Söderström Osakeyhtiö 2006).

Bruegge B and Dutoit AH, *Object–Oriented Software Engineering. Using UML, Patterns, and Java* (Pearson 2014).

Harenko K, Niiranen V, and Tarkela P, *Tekijänoikeus. Kommentaari ja käsikirja* (Talentum 2006).

Hervey T and others, Legal research methodologies in EU and international law (Hart Publishing 2011).

Husa J, *Oikeusvertailu* (Lakimiesliiton kustannus 2013).

McKusick MK, “Twenty years of Berkeley Unix: From AT&T-owned to freely redistributable” [1999] Open Sources: Voices from the Open Source Revolution 31.

Ojanen T, *EU-oikeuden perusteita* (Edita Publishing Oy 2016).

Pila J and Torremans P, *European intellectual property law* (Oxford University Press 2016).

Robert E, *Order without Law–How Neighbors Settle Disputes* (Harvard University Press 1991).

Schwab K, *The fourth industrial revolution* (Currency 2017).

Stallman R, The GNU Operating System and the Free Software Movement. Open Sources: Voices from the Open Source Revolution. C. Dibona, S. Ockman and M.

Stone. Calif (O'Reilly 1999) ⟨http://www.oreilly.com/catalog/opensources/book/stallman.html⟩.

Välimäki M and others, *The rise of open source licensing: a challenge to the use of intellectual property in the software industry* (Helsinki University of Technology 2005).

Välimäki M, *Oikeudet tietokoneohjelmistoihin* (Talentum 2009).

Vigna P and Casey MJ, *The age of cryptocurrency: how bitcoin and the blockchain are challenging the global economic order* (Macmillan 2016).

Williams S, *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002) ⟨https://www.oreilly.com/openbook/freedom/⟩.

Web sources

"2017 State of Linux Kernel Development" (*Linux foundation*) ⟨https://www.linuxfoundation.org/2017-linux-kernel-report-landing-page/⟩ accessed 31 May 2020.

"Apple Public Source License 2.0" (*Open Source Initiative*) ⟨https://opensource.org/licenses/APSL-2.0⟩ accessed 10 July 2020.

"Artistic License 2.0" (*Open Source Initiative*) ⟨https://opensource.org/licenses/Artistic-2.0⟩ accessed 23 May 2020.

"Berne Convention contracting parties" (*World Intellectual property organization*) ⟨https://www.wipo.int/treaties/en/ShowResults.jsp?treaty_id=15⟩ accessed 23 May 2020.

"Bill Gates: A timeline" (*BBC News*, 15 June 2006) ⟨http://news.bbc.co.uk/2/hi/business/5085630.stm⟩ accessed 5 August 2020.

"Bitcoin Energy Consumption Index" (*Digiconomist*) ⟨https://digiconomist.net/bitcoin-energy-consumption⟩ accessed 28 July 2020.

"Class Action" (*Cambridge Dictionary*) ⟨https://dictionary.cambridge.org/dictionary/english/class-action⟩ accessed 30 May 2020.

"Common Public License, version 1.0" (*Open Source Initiative*) ⟨https://opensource.org/licenses/cpl1.0.php⟩ accessed 10 July 2020.

"Copyright assignment at the FSF" (*Free Software Foundation*) ⟨https://www.fsf.org/bulletin/2014/spring/copyright-assignment-at-the-fsf⟩ accessed 31 May 2020.

"Developer Survey Results 2018" (*Stackoverflow*) ⟨https://insights.stackoverflow.com/survey/2018/#work-_-version-control⟩ accessed 6 June 2020.

"Free Software Foundation" ⟨https://fsf.org⟩ accessed 23 May 2020.

"git-push documentation" (*Git documentation*) ⟨https://git-scm.com/docs/git-push#Documentation/git-push.txt--f⟩ accessed 6 June 2020.

"GNU General Public License, version 1" ⟨https://www.gnu.org/licenses/old-licenses/gpl-1.0.html⟩ accessed 23 May 2020.

"GNU General Public License, version 3" ⟨https://www.gnu.org/licenses/gpl-3.0.html⟩ accessed 23 May 2020.

"GNU Software" ⟨https://www.gnu.org/software⟩ accessed 23 May 2020.

Haff G, "The mysterious history of the MIT License" (*Opensource*, 26 April 2019) ⟨https://opensource.com/article/19/4/history-mit-license⟩ accessed 23 May 2020.

"History of the OSI" (*Open Source Initiative*) ⟨https://opensource.org/history⟩ accessed 23 May 2020.

"History of the OSI" (*Open Source Initiative*) ⟨https://opensource.org/osd⟩ accessed 23 May 2020.

"Mozilla Public License 2.0" (*Open Source Initiative*) ⟨https://opensource.org/licenses/MPL-2.0⟩ accessed 9 August 2020.

"Nokia Open Source License Version 1.0a" (*Open Source Initiative*) ⟨https://opensource.org/licenses/Nokia⟩ accessed 10 July 2020.

"Open source license usage on GitHub.com" (*The GitHub Blog*) ⟨https://github.blog/2015-03-09-open-source-license-usage-on-github-com/⟩ accessed 10 July 2020.

"Popular Licenses" (*Open Source Initiative*) ⟨https://opensource.org/licenses/⟩ accessed 30 May 2020.

Stallman RM, *Initial announcement of the gnu project* (1983) ⟨http://www.gnu.org/gnu/initial-announcement.html⟩.

"The MIT License" (*Open Source Initiative*) ⟨https://opensource.org/licenses/MIT⟩ accessed 23 May 2020.

Case law

Finnish case law

KKO 1989:151.

KKO 1998:91.

KKO 1999:151.

KKO 2004:18.

MAO:285/19.

EU case law

Case 26/62 *NV Algemene Transport- en Expeditie-Onderneming van Gend en Loos v Nederlandse Administratie der Belastingen* [1963] ECR.

Case 6/64 *Flaminio Costa v ENEL* [1964] ECR 585.

Case 106/77 *Simmenthal* [1978] ECR 629.

Case 14/83 *Sabine von Colson and Elisabeth Kamann v Land Nordrhein-Westfalen* [1984] ECR.

Case 393/09 *Bezpečnostní softwarová asociace - Svaz softwarové ochrany v Ministerstvo kultury* [2009] ECR.

Case C-5/08 *Infopaq International A/S v Danske Dagblades Forening* [2009] ECR.

Case 406/10 *SAS Institute Inc v World Programming Ltd,* [2012] ECR.

Case 666/18 *IT Development SAS v Free Mobile SAS* [2019] ECR.

US case law

Jacobsen v Katzer 609 F Supp 2d 925 (ND Cal 2009).

Legislation

Finnish Legislation

Act on Class Actions, 444/2007.

Copyright Act, 404/1961.

Tort Liability Act, 412/1974.

EU Legislation

Corrigendum to Directive 2004/48/EC of the European Parliament and of the Council of 29 April 2004 on the enforcement of intellectual property rights [2004] OJ L195/ 16.

Directive 2001/29/EC of the European Parliament and of the Council of 22 May 2001 on the harmonisation of certain aspects of copyright and related rights in the information society [2001] OJ L167/10.

Directive 2006/116/EC of the European Parliament and of the Council of 12 December 2006 on the term of protection of copyright and certain related rights [2006] OJ L372/12.

DIRECTIVE 2009/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 April 2009 on the legal protection of computer programs [2009] OJ L111/16.

Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases [1996] OJ L77/20.

International treaties

Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3.

International Covenant on Economic, Social and Cultural Rights 993 UNTS 3.

Rome Convention for the Protection of Performers, Producers of Phonograms and Broadcasting Organizations 496 UNTS 43.

The Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) (1994) 1869 UNTS 299.

WIPO Copyright Treaty (1996) 2186 UNTS 121.

Abbreviations

ECJ	European Court of justice
EU	European Union
InfoSoc	Information Society Directive
KKO	Finnish supreme court
MAO	Finnish market court
WIPO	World Intellectual Property Organization

1 Introduction

1.1 Background

Open-source software projects are software which developers have published under open source licenses in a way the source code of the software is available for everyone. Commonly, contributing these projects is a hobby for its contributors. On the other hand, there are also projects where companies pay their employees who contribute projects.

The importance of open source licenses has grown over time. The reason for the importance of open source is that more and more people contribute to the open-source software project. There are also more cases for open source software than ever before. The empiric research has revealed there are three main reasons for the contribution.[1]

The first reason for contributing open-source software project is that contributing is directly beneficial to either contributor or its employer. For instance, a developer can contribute software development tools, making his or her work more comfortable. That

[1]Shaosong Ou Alexander Hars, "Working for free? Motivations for participating in open-source projects" (2002) 6(3) International Journal of Electronic Commerce 25; Josh Lerner and Jean Tirole, "Some simple economics of open source" (2002) 50(2) The journal of industrial economics 197; Guido Hertel, Sven Niedner, and Stefanie Herrmann, "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel" (2003) 32(7) Research policy 1159; Karim R Lakhani and Eric Von Hippel, "How open source software works:"free" user-to-user assistance" in *Produktentwicklung mit virtuellen Communities* (Springer 2004).

way contributing open source project makes contributors life easier many ways.[2]

The second reason for contributing open-source software project is that contributors see contributing as an opportunity to learn new skills. It is easy to see contributing open-source software teach new skill because, in any project, there is an instance which checks all contributions and accepts or decline them depend on quality. In this process, contributors get feedback about his or her contributions and learn.[3]

The third reason for contributing open-source software project is that contributors present software development skills in the form of the contribution and hope employers find him or her that way. As many companies use open-source software today, contributing them increase remarkable chances to become noticed. Increased attention for the contributor is because an employer can see who is a coder in a certain part of the source code and also what is the quality of code in this part. That way, contributors with great quality of code can become recruited, and the employer knows right away

[2]Shaosong Ou Alexander Hars, "Working for free? Motivations for participating in open-source projects" (2002) 6(3) International Journal of Electronic Commerce 25; Josh Lerner and Jean Tirole, "Some simple economics of open source" (2002) 50(2) The journal of industrial economics 197; Guido Hertel, Sven Niedner, and Stefanie Herrmann, "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel" (2003) 32(7) Research policy 1159; Karim R Lakhani and Eric Von Hippel, "How open source software works:"free" user-to-user assistance" in *Produktentwicklung mit virtuellen Communities* (Springer 2004).

[3]Shaosong Ou Alexander Hars, "Working for free? Motivations for participating in open-source projects" (2002) 6(3) International Journal of Electronic Commerce 25; Josh Lerner and Jean Tirole, "Some simple economics of open source" (2002) 50(2) The journal of industrial economics 197; Guido Hertel, Sven Niedner, and Stefanie Herrmann, "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel" (2003) 32(7) Research policy 1159; Karim R Lakhani and Eric Von Hippel, "How open source software works:"free" user-to-user assistance" in *Produktentwicklung mit virtuellen Communities* (Springer 2004).

how skilled programmers they are.[4]

Another part of the popularity of open-source software is that many software companies have moved to use open source components in their products. The reasons for the increasing popularity of open-source software in the software companies are efficiency and better quality of the code. It is efficient for companies to use open source because they do not have to pay all software development expenses in that case. There may already be some open-source project which satisfies the needs of the company, and therefore, there is no need for developing the company's own solution. On the other hand, if there is no suitable open source solution available, it can establish its open-source project. After the establishment of the open-source, it is possible also developers who do not work in the company start to contribute to the project.[5]

On the other hand, the quality of the code increases when more developers find possible bugs from code. It is easy to see that more developers see more errors in code. The quality of the code also increases because when more developers are involved in the project, the probability that someone invents a way to make the program more efficient.[6]

Using open-source licenses can also be problematic for companies because the licenses

[4]Shaosong Ou Alexander Hars, "Working for free? Motivations for participating in open-source projects" (2002) 6(3) International Journal of Electronic Commerce 25; Josh Lerner and Jean Tirole, "Some simple economics of open source" (2002) 50(2) The journal of industrial economics 197; Guido Hertel, Sven Niedner, and Stefanie Herrmann, "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel" (2003) 32(7) Research policy 1159; Karim R Lakhani and Eric Von Hippel, "How open source software works:"free" user-to-user assistance" in *Produktentwicklung mit virtuellen Communities* (Springer 2004).

[5]Joel West, "How open is open enough?: Melding proprietary and open source platform strategies" (2003) 32(7) Research policy 1259.

[6]Joel West, "How open is open enough?: Melding proprietary and open source platform strategies" (2003) 32(7) Research policy 1259.

require that source code is published. The requirement of source code publication is the reason why many companies also use a hybrid strategy. In this strategy, part of the company's projects are open source, but some are under proprietary licenses. That way the company collects benefits from open source but try to keep control using proprietary licenses in their end products.[7]

On the other hand, also the number of legal problems has increased over time.[8] The situation with open source licenses was long in such that many court judgments were not related to open source licenses. The reason for the lack of the judgment was that most open source projects were hobby projects, and nobody had time and interest to control who use of the end product of the hobby project and what way. And if there was a dispute, these disputes where settled because there was no case law about whether open-source licenses are legally binding in the court.[9]

It was unclear whether or not these licenses were contracts or intellectual property licenses. Now there are judgments both from the EU member states[10] and the US[11], which makes it clear that open source licenses can be both contract and intellectual property license. Although this problem is solved, there are still some problems unsolved. Because there are many contributors to open source projects, there are problems regarding who can raise a claim when somebody breaches the license.

[7] Joel West, "How open is open enough?: Melding proprietary and open source platform strategies" (2003) 32(7) Research policy 1259.

[8] Jennifer Buchanan O'Neill and Christopher J Gaspar, "What Can Decisions by European Courts Teach Us About the Future of Open-Source Litigation in the United States" (2010) 38 AIPLA QJ 437, p. 444.

[9] Heather J Meeker, "Open Source and the Age of Enforcement" (2012) 4 Hastings Sci. & Tech. LJ 267, p. 268.

[10] Armijn Hemel and Shane Martin Coughlan, "Making Sense Of Git In A Legal Context" (2017) 9 IFOSS L. Rev. 19, p. 19.

[11] Heather J Meeker, "Open Source and the Age of Enforcement" (2012) 4 Hastings Sci. & Tech. LJ 267, p. 269.

1.2 Research question and scope limitation

In this research, the question is that if open-source licenses are enforceable, who can start the process against the party, which breaches the license. Capability to start the process is an interesting problem because, typically, many persons contribute to the open-source project. While multiple software developers contribute to the project raises the subquestion about whether these contributors can raise a claim as a group or must everyone raise a claim as an individual.

It is also possible that these contributors have entirely different interests regarding the open-source software project. The changes are that most contributors do not have time or other resources to participate in the legal process. Can somebody participate on behalf of them?

The study's goal is to analyze, especially those who are the right owners in an open-source software project, and their relationship with each other. It is essential to understand how complex copyright ownership can go when the number of contributors increases. The literature has not much covered that question.

Problems can occur, for instance, if many developers contribute to open-source software. Suppose some party breaches a license who raise a claim. The question is not easy because copyright ownership is not always clear, and there can even be different opinions about whether raising a claim is necessary among the developers.

One possibility is that everyone who has ever contributed the project raises a claim. In this case, the question can be whether every contributor even has the copyright to the project. An individual developer's contribution can be too little or trivial to establish the right to raise a claim about copyright infringement. Another practical problem can be some contributors, who have widely contributed the project in the past, do not contribute to the project anymore, and other contributors cannot contact these contributors.

Another scenario is that one copyright owner can raise a claim on behalf of the whole project. Claiming behalf of others is possible if it considers the open-source software project as jointly authored work where it is not possible to identify separate contributors' contributions from each other. This approach makes the process more comfortable, but it requires some way to distribute money to contributors equally. It can be practically hard to contact all contributors and maybe even harder to establish consensus about money distribution.

Open source license enforcement is an exciting topic because, in many companies, open-source software compliance is not proper. In this situation, it is essential to know who can raise claims, or can anyone, who has ever contributed open-source software project, raise claim behalf all contributors.

The second research question, which has a clear connection to the first is what is the status of open-source software in the copyright sense. Is it jointly authored work or collective work? This question is fundamental when determining contributors' rights to raise a claim.

Because the question is processual and process rules are different in every country, this research mostly covers Finnish legislation. Occasionally also US legislation is considered as a preference. The EU copyright legislation related to the research question is also covered as Finland is a member state of EU and therefore its legislation is binding in Finland.[12]

This research discovers only open-source software copyrights, although there are open-

[12]the EU legislation is even superior when compared to Finnish legislation. See Case 6/64 *Flaminio Costa v ENEL* [1964] ECR 585; Case 106/77 *Simmenthal* [1978] ECR 629

source licensing in other types of work, which copyrights can protect.[13] Although the fundamental idea about open-source is similar for all works, working processes can be different, and henceforward analyzing all is too complex for this research.

1.3 Methodology

The method for this research is the source of law doctrine. The main focus is on Finnish and EU legislation. Order of the source is adapted from Aulis Aarnio because his source of law doctrine is the standard way to systematize Finnish sources of law. In his book 'Tulkinnan Taito'[14], Aarnio presents three different types of norms. The highest level is strongly binding sources.[15] Strongly binding sources must always take into account if they apply to the legal problem. These sources are National legislation and EU legislation. Aarnio also states that same European court justice(ECJ) judgment are strongly binding, but others are not.[16]

The next group of the norms is weakly binding norms. Norms in this group have interpretation power in solving the legal problem but not as much as strongly binding norms. Weakly binding norms are the legislator's aim, national case law, and these ECJ judgments, which are not strongly binding.[17]

The last category is the accepted sources. These sources are such that it is not punishable to use them in interpretation. Accepted sources are, e.g., legal literature.[18]

[13]e.g. Creative common licenses for visual works and TAPR for open-source hardware.

[14]Aulis Aarnio, *Tulkinnan taito* (Werner Söderström Osakeyhtiö 2006).

[15]Aarnio introduced his source of law doctrine in his book 'Laintulkinnan teoria'Aulis Aarnio, *Laintulkinnan teoria* (Werner Söderström Osakeyhtiö 1988) in 1982, but this doctrine does not include EU legislation since Finland was not a member state at that time.

[16]Aulis Aarnio, *Tulkinnan taito* (Werner Söderström Osakeyhtiö 2006).

[17]Aulis Aarnio, *Tulkinnan taito* (Werner Söderström Osakeyhtiö 2006).

[18]Aulis Aarnio, *Tulkinnan taito* (Werner Söderström Osakeyhtiö 2006).

Although the background of Aulis Aarnio's source of law doctrine is sort law sources by how punishable it is for a judge to not use the particular source in his or her judgment process, this doctrine is suitable for legal research too. The difference is the legal research search general solution when judge only tries to make the judgment in one specific case. Besides, legal research is not bound to current legislation, and argumentation mentioned above. The researcher can also use other argumentation and make statements about how current legislation should be changed.[19]

Because the ECJ case law's hierarchic status is quite confusing in Aarnio's theory, This research use for the relation between European Union jurisdiction and national law systematization which Tuomas Ojanen have expressed it in *EU-oikeuden perusteita*[20]. According to Ojanen, there are three principles which control the relation between national and EU jurisdiction. They are the primacy of European Union law, the direct effect of European Union law and the indirect effect of European Union law.[21]

The primacy of European Union law is a principle which states that, if there is a conflict between national legislation of the member state and European Union legislation, European Union legislation overrule national legislation. The primacy of European Union law principle is recognized in ECJ case law, and the first case where is applied is *Costa v. ENEL.*[22]

According to the direct effect of European Union law principle, EU legislation is applicable in the national court of the member state even when the member state has failed to implement it to the national legislation. The principle states EU legislation is applicable not only horizontally between member state and individual but also verti-

[19]Aulis Aarnio, *Laintulkinnan teoria* (Werner Söderström Osakeyhtiö 1988).

[20]Tuomas Ojanen, *EU-oikeuden perusteita* (Edita Publishing Oy 2016).

[21]Tuomas Ojanen, *EU-oikeuden perusteita* (Edita Publishing Oy 2016) p. 66.

[22]Case 6/64 *Flaminio Costa v ENEL* [1964] ECR 585; Tuomas Ojanen, *EU-oikeuden perusteita* (Edita Publishing Oy 2016) p. 86

cally between two individual in the member state. The development of the direct effect principle started from the *Van Gend & Loos* case.[23]

The undirect effect of European Union law is the principle which states that national court should interpret national jurisdiction of the member state with the aim that it is in line with EU legislation. The case which started the formation of the principle was *Von Colson*[24].

Although the source of laws doctrine is the primary method for this research, the research use also the comparative law method in some situations. As Jaakko Husa states in his book *Oikeusvertailu*, comparative law is a flexible term, but common to all comparative law is that it has a target and compares jurisdiction between two or more countries.[25] This research compares Finnish and US jurisdiction, which related to open-source licenses and copyright. The aim of that is to use the US as a reference. This kind of method is not demanding in theoretical comparative law sense but give a broader view about the topic.[26]

The reason for using comparative law as a supportive method is that many open source projects are contributed worldwide, especially from the EU and the US. Necessarily it is essential to determine how these two jurisdictions differ from each other when it comes to enforcing copyright licenses in the open-source context. US legislation is essential also because many information technology companies have headquarters there.[27]

[23]Case 26/62 *NV Algemene Transport- en Expeditie-Onderneming van Gend en Loos v Nederlandse Administratie der Belastingen* [1963] ECR; Tuomas Ojanen, *EU-oikeuden perusteita* (Edita Publishing Oy 2016) p. 72

[24]Case 14/83 *Sabine von Colson and Elisabeth Kamann v Land Nordrhein-Westfalen* [1984] ECR; Tuomas Ojanen, *EU-oikeuden perusteita* (Edita Publishing Oy 2016) p. 91

[25]Jaakko Husa, *Oikeusvertailu* (Lakimiesliiton kustannus 2013).

[26]Jaakko Husa, "Valkoista yksisarvista pyydystämässä vai mörköä paossa – »oikeaa oikeusvertailua»?" (2010) 2010(5) Lakimies 700, p. 702.

[27]For instance Microsoft HQ is located in Seattle, and Facebook HQ is in California.

As the law is not a separate part of the society, this thesis uses socio-legal method[28] to discover how the matters change when discovering the phenomenon from a software engineering perspective compared to the legal perspective. The sociological method is adopted to research because open source development is a software engineering process. This process has an impact on the copyrights of the end product of the open-source project. For this reason, it is essential to look at the situation also from the software developer perspective.

1.4 Structure

The structure of the research will be such that first, it explains open-source licenses as a legal construction. The explanation includes a description of the history of open-source licenses and an introduction about the most used types of open source licenses. The chapter aims to explain open-licenses as a legal concept that the reader is familiar with the concept.

Chapter three of this writing describes software development both non-open source and open source perspective. The chapter cover phases of the software development process and also how they are different in open source development. In addition to introducing the software development process, the chapter introduces the roles of the people involved in the development process. The aim in this chapter is to get understanding about the process which generate the open-source software.

After that, chapter four describes copyright as a legal concept. The chapter will introduce copyright legislation from the international, European Union, and the Finnish national level. Especially that chapter discover legislation which is related to the com-

[28]Tamara Hervey and others, Legal research methodologies in EU and international law (Hart Publishing 2011) p. 86.

puter program.

Chapter five discover how to find authors of the code in open source projects and determine which are their contributions. Decentralized version control system Git is also introduced in this chapter as it is one possible answer for authorship problems. Also, the solution which uses blockchain is introduced, and its benefits and challenges are determined.

After that, the research will describe the condition for open-source licenses to be enforceable. In this chapter, relevant legislation and case law are covered both from Finland and the EU level. It is also discovered who generally raises copyright claims if there more than one author for work. The last chapter will conclude.

2 Definition of open source

2.1 History

The story of Open source licenses starts from early 1980s[29]. Furthermore, it all started with printer[30]. Alternatively, more preciously from printing jam in the printer in Massachuset Institute of Technology Artificial Intelligence Lab where Richard Stallman worked at the time. From the printing jam started the chain of events that established foundations for Open Source movement.

At the beginning of the 1980s, there was no personal computer in the same way as currently. Big central computers performed nearly all electronic information processing, and they could take tens of cube meters space. In addition to those central computers, there was also a terminal that had no other function than providing access to the central computer. Terminals made it possible for multiple persons can use the computational power of the central computer at the same time.[31]

Because central computers were costly and electronically complex systems, the manufacturer's primary focus was on the computer's electronic side, not on the software side. Therefore hardware was the main component of the product, and companies provided

[29]In addition to the following thread of history, which is related to Richard Stallman, there is also a branch in this story which roots are in Berkeley Software Distribution. This story is told in Marshall Kirk McKusick, "Twenty years of Berkeley Unix: From AT&T-owned to freely redistributable" [1999] Open Sources: Voices from the Open Source Revolution 31

[30]Sam Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002) ⟨https://www.oreilly.com/openbook/freedom/⟩.

[31]Sam Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002) ⟨https://www.oreilly.com/openbook/freedom/⟩.

the human-readable source code of the software free because there was not yet an idea about the business value of the software[32]. Or there were some first software firms already, but that industry just took its first steps. For instance, Bill Gates and Paul Allen founded Microsoft already in 1975 and released its first software product in the same year.[33]

At the beginning of the 1980s, printer jams were frequent. Printer jams were a problem because when the person sent documents to the printer, it was not possible to know whether the printer had jammed. When the person went deliver his prints, it could be that they were not ready because of jam. These printing jams cause a lot of frustration and unnecessary walking back and forth around the printer.[34]

To solve the problem with printer jams, Richard Stallman, who worked in MIT AI Labs, modified the software of the printer of the MIT AI Labs such that in case of a printer jam, it sent each person who had works in print queue message. In such wise, anyone who wanted to receive a printout would then know to fix the problem. Sending information about printing jams was possible because Stallman had access to source code of the printer software. With the source code, Stallman could make needed modifications to the software to make it notify about the jam.[35]

Stallman's fix for the printer jams worked until MIT AI Lab received a new Xerox printer, which the manufacturer did not provide source code with it. Although the new

[32]Sam Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002) ⟨https://www.oreilly.com/openbook/freedom/⟩.

[33]"Bill Gates: A timeline" (*BBC News*, 15 June 2006) ⟨http://news.bbc.co.uk/2/hi/business/5085630.stm⟩ accessed 5 August 2020.

[34]Sam Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002) ⟨https://www.oreilly.com/openbook/freedom/⟩.

[35]Sam Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002) ⟨https://www.oreilly.com/openbook/freedom/⟩.

printer was more sophisticated than the previous one, it still had the same problem with jamming papers. This time difference was that Stallman could not fix the jamming problem because the source code was not available. Without the source code, it was not possible to modify the software.[36]

The missing source code was why Stallman contacted the professor who Stallman knew participated development process of the printer. The professor could not help because he had signed a non-disclosure agreement about software in the printer. The moment when Stallman heard about non-disclosure agreement was the turning point in which Richard Stallman realized that the software development industry was in change. Or the value of the software had become more and more noticeable.[37]

Stallman could not accept that change. His opinion was that people should always share software because it is almost free to make a copy of the software. Stallman's ideology was the reason why he started to develop an operating system in which source code would be available for everyone. Stallman called the project "GNU" [38].[39] As time pass GNU project has grown to include multiple software. In addition to the operating system, the GNU project currently includes much other open-source software such as image manipulation program GIMP and scientific writing software TexMacs[40].

While Stallman worked with GNU, he also searched copyright licenses for the project. The license requirement was that it should maximize the sharing of and access the source code. It was a long process to find a suitable way to license the GNU software

[36]Sam Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002) ⟨https://www.oreilly.com/openbook/freedom/⟩.

[37]Sam Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002) ⟨https://www.oreilly.com/openbook/freedom/⟩.

[38]recursive acronym "GNU is not UNIX"

[39]Sam Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002) ⟨https://www.oreilly.com/openbook/freedom/⟩.

[40]"GNU Software" ⟨https://www.gnu.org/software⟩ accessed 23 May 2020.

because Stallman had strict criteria for the license. Stallman tested many different licenses for the project in its early days.[41] These experiments led to the evolution of GNU General Purpose License in February 1989.[42]

2.2 Definition of open source license

2.2.1 Free software Foundation

Free software Foundation (FSF) is a non-profit organization which promotes freedom of computer user. As a non-profit organization, most of the fund for FSF come for donations. Richard Stallman founded FSF in 1985 to promote free software especially his GNU project.[43]

FSF defines free software. The definition set four requirements for the software that it should fulfil to be free software. the definition states that requirements for free software are such that

- *"You have the freedom to run the program, for any purpose."*
- *"You have the freedom to modify the program to suit your needs. (To make this freedom effective in practice, you must have access to the source code, since making changes in a program without having the source code is exceedingly difficult.)"*

[41] Sam Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002) ⟨https://www.oreilly.com/openbook/freedom/⟩.

[42] Sam Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002) ⟨https://www.oreilly.com/openbook/freedom/⟩ Version 1 of the GNU GPL is available at "GNU General Public License, version 1" ⟨https://www.gnu.org/licenses/old-licenses/gpl-1.0.html⟩ accessed 23 May 2020

[43] see. "Free Software Foundation" ⟨https://fsf.org⟩ accessed 23 May 2020.

- "*You have the freedom to redistribute copies, either gratis or for a fee.*"
- "*You have the freedom to distribute modified versions of the program, so that the community can benefit from your improvements.*"[44]

The base of this definition is on Stallman's ideology. According to Stallman, there is "Golden rule" which is that programmer should always share his software.[45] The definition is ideological. Ideology can be seen in the wording of the definition, which is quite informal. Because of the ideological nature of the definition, there are not many requirements set at least when compared to the Open Source Initiative's definition, which is introduced in the next subsection.

2.2.2 The Open Source Initiave

When base FSF's definition has the ideological background, The Open Source Initiative (OSI) see more practical benefits in open source. OSI was founded in 1998, and its mission is to promote open source development process. OSI keep also track about open-source licenses. When compared to the FSF, OSI is much more commercially

[44]It is essential to make a difference between "free" as "freedom" and "free" as "cost." Free software is free in the former sense not in a latter sense Richard Stallman, The GNU Operating System and the Free Software Movement. Open Sources: Voices from the Open Source Revolution. C. Dibona, S. Ockman and M. Stone. Calif (O'Reilly 1999) ⟨http://www.oreilly.com/catalog/opensources/book/stallman.html⟩.

[45]Stallman wrote: I consider that the golden rule requires that if I like a program, I must share it with other people who like it. I cannot, in good conscience, sign a non-disclosure agreement or a software license agreement. So that I can continue to use computers without violating my principles, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free. Richard M Stallman, *Initial announcement of the gnu project* (1983) ⟨http://www.gnu.org/gnu/initial-announcement.html⟩ initially posted to net.unix-wizards Usenet group.

oriented organization.[46] OSI has a definition for open software license which include ten requirements which are

1. "***Free Redistribution***

   The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several sources. The license shall not require a royalty or other fee for such sale."

2. *"**Source Code***

   The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed".

3. *"**Derived Works***

   The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software."

4. *"**Integrity of The Author's Source Code***

   The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the

[46]see "History of the OSI" (*Open Source Initiative*) ⟨https://opensource.org/history⟩ accessed 23 May 2020.

source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software."

5. "***No Discrimination Against Persons or Groups***

   The license must not discriminate against any person or group of persons."

6. "***No Discrimination Against Fields of Endeavor***

   The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research."

7. "***Distribution of License***

   The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties."

8. "***License Must Not Be Specific to a Product***

   The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution."

9. "***License Must Not Restrict Other Software***

   The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must *be open-source software."*

10. ***"License Must Be Technology-Neutral***

    No provision of the license may be predicated on any individual technology or style of interface."[47]

This definition is more precise and practically-oriented than FSF's one. On the other hand, most of the licenses fulfil both definitions.[48] Also, the wording was more formal in the OSI definition than in the FSF definition. It is also essential to notice that free software definition speaks requirements for software when open source definition speaks requirements for licenses. Then again licenses which fulfil requirements set in open source definition get a certificate about it. That certification creates an end-users assumption that the license fulfils certain norms set by open-source community[49].

2.3 Different open-source licenses

There is two commonly used way to categorize open-source licenses; either based on their functionalities or their historical origin. Functionality based categorization classifies licenses by the content of the licenses. Then again, historical categorization use phases in the history of the open-source movement to set licenses in the categories. Functional categories are licenses with strong reciprocity obligations, licenses

[47]see. "History of the OSI" (*Open Source Initiative*) ⟨https://opensource.org/osd⟩ accessed 23 May 2020.

[48]Henrik Udsen, "Open source licences" in *User Generated Law* (Edward Elgar Publishing 2016) p. 104.

[49]Mikko Välimäki, "Avoimen lähdekoodin ohjelmistolisensseistä" (2002) 5 Defensor Legis, p. 854.

with standard reciprocity[50] obligations and permissive licenses. Copyleft license is the term to call licenses with strong and standard reciprocity obligations together. Then again based on historical origin licenses are grouped to GNU, academic, community and corporate licenses.[51]

Characteristic for the licenses with strong reciprocity obligations is that they require that licenses terms remain the same in the adaptive works and derivative works of the work which is licenses under a license which belongs to that category. Licenses with standard reciprocity obligations differ from the licenses with strong reciprocity obligations such that when the developer combine the work with standard reciprocity with other work changes in license terms are possible. That is not allowed with the strong reciprocity licenses. For instance, the General Public License is a license with strong reciprocity obligations.[52]

The main difference between copyleft licenses (licenses with standard and strong reciprocity obligations) and permissive licenses is the requirements they set to the licenses of the derivative works. Permissive licenses leave the author of the derivate work more freedom to choose what license he or she will use in his or her derivative work. That does not mean that a permissive license does not set any requirements for derivative works. For instance, Lesser General Public License is a license with strong reciprocity obligations.[53]

[50]It is possible also to use terms restrictive and highly restrictive licenses. see. Josh Lerner and Jean Tirole, "The scope of open source licensing" (2005) 21(1) Journal of Law, Economics, and Organization 20

[51]Mikko Välimäki and others, *The rise of open source licensing: a challenge to the use of intellectual property in the software industry* (Helsinki University of Technology 2005) p. 107.

[52]Mikko Välimäki and others, *The rise of open source licensing: a challenge to the use of intellectual property in the software industry* (Helsinki University of Technology 2005) p. 118.

[53]Mikko Välimäki and others, *The rise of open source licensing: a challenge to the use of intellectual property in the software industry* (Helsinki University of Technology 2005) p. 118.

In the historical categorization, the earliest license type is GNU licenses. Origin of these licenses is in the 1980s, and they have an entirely ideological background. The ideological background is the reason why the target audience for these licenses is other developers and wording reflects the audience. GNU licenses have been criticized about incompatibility problems with many other open-source licenses which are at least partly caused by Richard Stallman's ideological choices.[54]

Second historical type of open-source licenses is the academical licenses. Origin of these licenses is in the US universities and their contributions for the telecommunication. These licenses are typically easy to read for also other than the developer. They are also permissive licenses and compatible with other open-source licenses. One example about academic license is MIT license.[55]

Third historical type of open-source licenses is community license. Typically these licenses are written for some free software project. Typically these projects have been related to Internet or UNIX implementations. The wording in these licenses vary much as Artistic License which is created for Perl programming language contains much terminology which is ambiguous. On the other hand, Apache Software license which is also community license is more clear.[56]

The fourth category is corporate licenses. Companies who maintain open-source projects have drafted these licenses. It is typical for the corporate licenses that they have

[54]Mikko Välimäki and others, *The rise of open source licensing: a challenge to the use of intellectual property in the software industry* (Helsinki University of Technology 2005) p. 120.

[55]Mikko Välimäki and others, *The rise of open source licensing: a challenge to the use of intellectual property in the software industry* (Helsinki University of Technology 2005) p. 120.

[56]Mikko Välimäki and others, *The rise of open source licensing: a challenge to the use of intellectual property in the software industry* (Helsinki University of Technology 2005) p. 121.

many details and legalese.[57] Examples about corporate licenses are Common Public License(IBM)[58], Apple Public Source License (Apple)[59] and Nokia Open Source License(Nokia)[60].

The next subsection introduces two permissive licenses (MIT and Artistic license) and one copyleft (GPL) license in more detail. MIT and GPL are introduced because they are viral licenses for open source projects[61]. Although an Artistic license is not that common, it is introduced because it has a remarkable role in case law related to the legal nature of the open-source licenses and described later in this writing.

2.3.1 General Public License

General Public License (GPL) is the license that is initially drafted for the GNU project. The licensee of the software license under GPL is allowed to make copies about the source code of the software if the license is added to the copy.[62]

It states that derivative works must be licensed under GPL. There shall also be a notification about modification and the date of modification. These requirements do

[57]Mikko Välimäki and others, *The rise of open source licensing: a challenge to the use of intellectual property in the software industry* (Helsinki University of Technology 2005) p. 121.

[58]see. "Common Public License, version 1.0" (*Open Source Initiative*) ⟨https://opensource.org/licenses/cpl1.0.php⟩ accessed 10 July 2020.

[59]see. "Apple Public Source License 2.0" (*Open Source Initiative*) ⟨https://opensource.org/licenses/APSL-2.0⟩ accessed 10 July 2020.

[60]see. "Nokia Open Source License Version 1.0a" (*Open Source Initiative*) ⟨https://opensource.org/licenses/Nokia⟩ accessed 10 July 2020.

[61]MIT and GPL was most and second most used licenses in the Github in 2015"Open source license usage on GitHub.com" (*The GitHub Blog*) ⟨https://github.blog/2015-03-09-open-source-license-usage-on-github-com/⟩ accessed 10 July 2020

[62]"GNU General Public License, version 3" ⟨https://www.gnu.org/licenses/gpl-3.0.html⟩ accessed 23 May 2020.

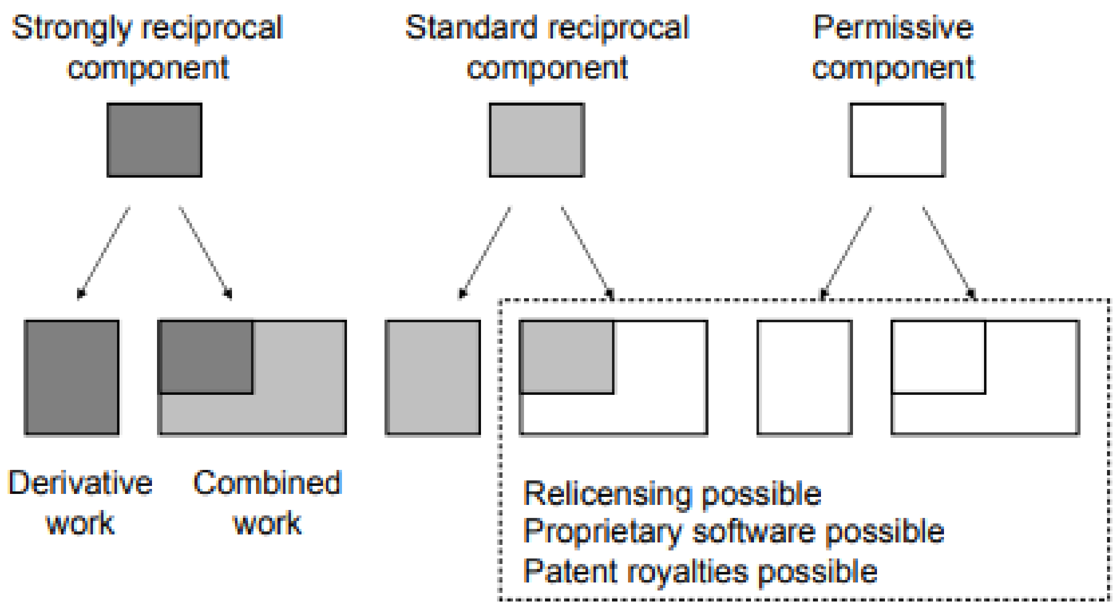

Strongly reciprocal component
Standard reciprocal component
Permissive component
Derivative work
Combined work
Relicensing possible
Proprietary software possible
Patent royalties possible

Figure 2.1: Functional differences regarding combination and modification between open source licenses (Mikko Välimäki and others, *The rise of open source licensing: a challenge to the use of intellectual property in the software industry* (Helsinki University of Technology 2005) p. 119)

not depend on how software is packed.[63]

In addition to copyright matters, GPL considers the possibility that some contributors file a patent regarding the software copyright licensed under GPL. For this situation GPL requires that the contributor provide *"non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version"*.[64] The clause prevents a patent owner from limiting rights providing in General Purpose License.

[63]"GNU General Public License, version 3" ⟨https://www.gnu.org/licenses/gpl-3.0.html⟩ accessed 23 May 2020.

[64]"GNU General Public License, version 3" ⟨https://www.gnu.org/licenses/gpl-3.0.html⟩ accessed 23 May 2020.

2.3.2 MIT license

As MIT license name refers, it is from the Massachusets Institute of Technology. It is not clear what was an original use case for the MIT license, but it is thought that it is created for the X Window System in the 1980s.[65] This license gives the licensee the right to do nearly anything they want with the software.

The only exception it that licensee is not allowed to sue right holder about anything related to software, and there is no warranty that software works as expected.[66] This provision is in line with common sense as an average person would not complain about something they have got free.

2.3.3 Artistic license

Like MIT license, also an Artistic license is a permissive license. Originally Artistic license was a license for Perl scripting language. The aim of the Artistic license is for the author of the software to keep artistic control about the development of the software while the software is free and open-source. Artistic license allows redistribution of original work either gratis or with distribution fee.[67]

The artistic license does not set any limitations for derivative works if these derivative works are not distributed. On the other hand, the license set requirements for derivative works if the author of the derivative work distributes derivative work. The requirements

[65]Gordon Haff, "The mysterious history of the MIT License" (*Opensource*, 26 April 2019) ⟨https://opensource.com/article/19/4/history-mit-license⟩ accessed 23 May 2020.

[66]see. "The MIT License" (*Open Source Initiative*) ⟨https://opensource.org/licenses/MIT⟩ accessed 23 May 2020.

[67]see. "Artistic License 2.0" (*Open Source Initiative*) ⟨https://opensource.org/licenses/Artistic-2.0⟩ accessed 23 May 2020.

are that developer of the derivative work shall document changes he or she has made and do at least one of the following:

1. *"make the Modified Version available to the Copyright Holder of the Standard Version, under the Original License, so that the Copyright Holder may include your modifications in the Standard Version."*
2. *"ensure that installation of your Modified Version does not prevent the user installing or running the Standard Version. In addition, the Modified Version must bear a name that is different from the name of the Standard Version.*
3. *allow anyone who receives a copy of the Modified Version to make the Source form of the Modified Version available to others under"*
   (a) *"the Original License or"*
   (b) *"a license that permits the licensee to freely copy, modify and redistribute the Modified Version using the same licensing terms that apply to the copy that the licensee received, and requires that the Source form of the Modified Version, and of any works derived from it, be made freely available in that license fees are prohibited but Distributor Fees are allowed."*[68]

Above mentioned provision is near to copyleft, but option 2 makes it possible to do derivative software and freely choose the license for it. That is why the artistic license is a permissive license.[69]

[68]see. "Artistic License 2.0" (*Open Source Initiative*) ⟨https://opensource.org/licenses/Artistic-2.0⟩ accessed 23 May 2020.

[69]see. "Artistic License 2.0" (*Open Source Initiative*) ⟨https://opensource.org/licenses/Artistic-2.0⟩ accessed 23 May 2020.

3 Software engineering and software development process

3.1 Phases of open source development

The software development process contains five phases: requirements elicitation, analysis, module design, implementation, and testing.[70] Aim of the process is to produce the software which meets the set requirements as well as possible. It depends on used development methodology whether these phases are in a row or partly side by side.[71]

Requirements elicitation is a state where developer and end-user decide together what are requirements for the software. Typically these requirements are either functional or non-functional. Functional requirements are about the features which are related to how the software works. For instance, the functional requirement can be how the software reacts to the error or where some button is in the user interface[72]. Non-functional requirements are requirements that consider other parts of the software than functional. These requirements can be how fast software should fetch data from the database or how long it can take to start software.[73]

[70]Bernd Bruegge and Allen H Dutoit, *Object–Oriented Software Engineering. Using UML, Patterns, and Java* (Pearson 2014) p. 14.

[71]in waterfall model phase is entirely in a row and in scrum the software is initially divided to the parts, and that is why same part can be in analysis phase when other is in implementation phase

[72]user interface is the part of the software with which user can interact. Typically it contains buttons and text

[73]Bernd Bruegge and Allen H Dutoit, *Object–Oriented Software Engineering. Using UML, Patterns, and Java* (Pearson 2014) p. 14.

In the analysis phase requirement generated in requirements, elicitation is converted to a technical plan of how software should be organized. For instance, it is designed how software is divided into different modules. Also, the programming language is decided in this phase is that it is not defined in requirements elicitation phase.[74]

The implementation phase is where the actual code happens. In this phase, the software is developed based on documents that are from previous phases.[75] That is the phase where all source code is written.

The last phase is testing. In this phase, it is tested whether software fulfils requirements that are decided with the client in requirements elicitation. If the software passes the test, then the software is delivered to the client. Then again if tests fail it returns to the phase which is responsible for the fail and development is continue until the software is again in the testing phase.[76]

3.2 Software development roles

There are at least four different categories for the people in the software development project. These categories are Management roles, development roles, cross-functional roles, and consultant roles. A person who is in a management role, for instance, project manager, tries to organize software development to keep it in budget and schedule. A

[74]Bernd Bruegge and Allen H Dutoit, *Object–Oriented Software Engineering. Using UML, Patterns, and Java* (Pearson 2014) p. 14.

[75]Bernd Bruegge and Allen H Dutoit, *Object–Oriented Software Engineering. Using UML, Patterns, and Java* (Pearson 2014) p. 18.

[76]Bernd Bruegge and Allen H Dutoit, *Object–Oriented Software Engineering. Using UML, Patterns, and Java* (Pearson 2014) p. 18.

person with a development role takes part in the actual software development.[77]

Cross-functional roles are for communication from one team to others, keep contact with end-users, and establishing interfaces. Consultants bring temporary knowledge about fields in which the core team does not have expertise. This knowledge can be technical, for instance, about new programming paradigm, or non-technical, for instance, legal advice.[78]

3.3 Open source software development

For understanding the challenges in open source licenses, it is fundamentally important to understand how open-source software development process works. The open-source software development process happens most likely on Internet.[79] It is also good to notice that some person who participates in the project do it voluntarily, and as a hobby, in which case, for instance, their motivation influence their contribution.

Characteristic for the open-source software development is that amount of developers can be huge. On the other hand, also, the amount of changes in the team is significant over time. The typical behaviour is that developers participate in the project when it has a good reputation, and it is well known (so-called band-wagon effect). On the contrary, developers leave projects which are not successful or attractive in another way. Leaving developers is a challenge for development because some essential knowledge

[77]Bernd Bruegge and Allen H Dutoit, *Object–Oriented Software Engineering. Using UML, Patterns, and Java* (Pearson 2014) p. 82.

[78]Bernd Bruegge and Allen H Dutoit, *Object–Oriented Software Engineering. Using UML, Patterns, and Java* (Pearson 2014) p.82.

[79]Keng Siau and Yuhong Tian, "Open Source Software Development Process Model: A Grounded Theory Approach" (2013) 21(4) Journal of Global Information Management (JGIM) 103, p. 108.

for the project can leave with the developer.[80]

In most of the cases, there is some instance who control the development of the open-source project.[81] This can be a non-profit organization. However, it is also possible that it is a company.[82] The purpose of this instance is to decide which modifications to the project's source code are accepted. On the other hand, the instance decides guidelines for development. For instance, the guideline can contain information about what features the project should contain.

Because the source code of the open-source project is available and open-source licenses give right for derivative works, it is always possible that projects forks. Forking means that one open-source project divides to the two projects because the maintainer of the original project does not accept certain modifications that a group of contributors sees essential, and that is why a new parallel software development project is established.

[80]Gregory Madey, Vincent Freeh, and Renee Tynan, "The open source software development phenomenon: An analysis based on social network theory" [2002] AMCIS 2002 Proceedings 247, p. 1810.

[81]for instance Mozilla Foundation for Mozilla browser, Apache Foundation for Apache software and Linux Foundation for Linux Kernel

[82]for instance, Google controls the TensorFlow machine learning library, which is released under the Apache License.

4 Legal frame of copyright

4.1 Copyright as a Intellectual property

Intellectual property rights are a group of exclusive rights which aim is to protect a person who has used his or her creativity to create something new. Intellectual property rights give its owner the power to control how the target of the right can be used. This kind protection is internationally recognized for instance in Article 15(1) (c) of the International Covenant on Economic, Social and Cultural Rights which states that the states which are parties in the Covenant *"recognize the right of everyone – (c) To benefit from the protection of the moral and material interests resulting from any scientific, literary or artistic production of which he is the author"*[83].

Like all other intellectual properties, copyright is an exclusive right that gives its owner the right to prevent others from doing something with its object. In the case of copyright, this object is authorial work or some other expressive work. Unlike many other intellectual properties, there is no need to register copyright anywhere.[84] The protection starts with the creation of the work. With this definition, the scope of the copyright is quite broad, and nearly all artistic works from videos to computer programs are under the scope of copyright protection.

There are copyright regulations in international, EU, and national level. The following subsections will describe which kind of legislation is related to copyright starting from the international level and moving through the EU legislation to Finnish national

[83]International Covenant on Economic, Social and Cultural Rights 993 UNTS 3, 15(1)(c).

[84]Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3, art 5.2.

copyright legislation. That order is chosen because international regulation create foundation to the copyright legislation, which is sharpened in the EU and national legislation.

4.1.1 International copyright treaties

Berne convention for Literary and Artistic Works[85] sets internationally general guidelines for copyright. There are some provisions which give signing states more space to regulate copyright when others are more strict. Berne convention was originally accepted in 1886, and it was revised in Paris in 1971. According to the World Intellectual Property Organization(WIPO), 171 states have ratified the convention at the time of writing.[86]

Berne convention states the term of copyright. According to the Convention minimum term of copyright is 50 years from the author's death. In a joint authorship case, when there are multiple creators for the work, the duration of the copyright protection is counted from the death of the last survivor. However, if the author is anonymized or pseudonym, the term of copyright is 50 years from the work's publishing date. As these are the minimum periods for the copyright-protection, states can decide to set longer-term but not shorter.[87]

The convention also defines these "literary and artistic" works, which are objects for copyright. The definition lists many different types of works, and therefore the scope

[85]Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3.

[86]"Berne Convention contracting parties" (*World Intellectual property organization*) ⟨https://www.wipo.int/treaties/en/ShowResults.jsp?treaty_id=15⟩ accessed 23 May 2020.

[87]Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3, art 7.

of copyright-protection is defined broadly. According to The convention, literary and artistic work can be

> *"every production in the literary, scientific and artistic domain, whatever may be the mode or form of its expressions, such as books, pamphlets and other writings; lectures, addresses, sermons and other works of the same nature; dramatic or dramatico-musical works; choreographic works and entertainments in dumb show; musical compositions with or without words; cinematographic works to which are assimilated works expressed by a process analogous to cinematography; works of drawing, painting, architecture, sculpture, engraving and lithography; photographic works to which are assimilated works expressed by a process analogous to photography; works of applied art; illustrations, maps, plans, sketches and three-dimensional works relative to geography, topography, architecture or science."*[88]

The Berne Convention set minimum right for copyright owner. The convention also divide the right to two category; economical rights and moral rights. Economical rights are rights which are related to the economical value of the work. Economical values are movable such that they can be transfered from one person to the other. The convention give rightowner six exclusive economical rights:

- The right of reproduction[89],
- The right of translation[90],

[88]Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3, 2(1).

[89]Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3, art 9.

[90]Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3, 8,11,11ter.

- the right of adaption[91],
- the right of public performance and communication to the public[92],
- The right of public recitation and communication to the public of recitation[93]
- optional right to obtain a share on the resale of a work of art[94].

In addition to economic rights, the convention also provides moral rights to the author. Moral rights are right, which are not movable, and these rights are more related to the artistic values of the work. The idea about moral right is to promote creator's rights as an artist. Moral rights are for instance right to claim authorship and right to prevent such modification to the work which would influence to the honour and reputation of the author.[95]

Also WIPO Copyright Treaty(WCT)[96] regulates copyrights in the international level. WCT is an especially important treaty for information technology because it states that software shall be protected with copyright[97]. WCT also states that the selection or arrangement of their contents in the database is protected with copyright[98]. On the

[91]Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3, arts 12,14.

[92]Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3, 11,11bis.

[93]Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3, 11ter.

[94]Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3, 14ter.

[95]Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3.

[96]WIPO Copyright Treaty (1996) 2186 UNTS 121.

[97]WIPO Copyright Treaty (1996) 2186 UNTS 121, art 4.

[98]WIPO Copyright Treaty (1996) 2186 UNTS 121, art 5.

other hand, data stored to the database is not by definition protected with copyright according to the treaty.

In addition to WCT and Berne Convention, also TRIPS agreement[99] regulate copyrights matters in countries which are member states in the World Trade Organization. Aim of the TRIPS is set minimum standards for the Intellectual properties in the WTO member states. As the World Trade Organization has drafted TRIPS agreement, the aim of the treaty is also to provide a similar condition for international trade in all member states of the World Trade Organization. TRIPS agreement also states that software is in the scope of copyright regulation.[100]

4.1.2 Copyright in European Union

In EU copyright is regulated with many directives. All of the directives regulate one specific aspect of the copyright, copyright protection in spefic circustances or copyright for specific type of work such that there is no general copyright directive. Most central of these directives is Information society directive (InfoSoc)[101] which defines scope of copyright protection in EU.[102] InfoSoc basically adapt Berne convention[103] and Rome

[99]The Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) (1994) 1869 UNTS 299.

[100]The Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) (1994) 1869 UNTS 299, art 10.

[101]Directive 2001/29/EC of the European Parliament and of the Council of 22 May 2001 on the harmonisation of certain aspects of copyright and related rights in the information society [2001] OJ L167/10.

[102]Justine Pila and Paul Torremans, *European intellectual property law* (Oxford University Press 2016) p. 243.

[103]Berne Convention for the Protection of Literary and Artistic Works, September 9, 1886, revised at Paris July 24, 1971 1161 UNTS 3.

convention[104] obligations to the EU legislation.[105] In addition to InfoSoc, Copyrights are regulated in Computer Program directive[106], term directive[107] and Database directive[108]. Computer Program directive is covered more extensively in section 4.2 and other directives narrowly in this section.

The term directive is about how long copyright protection is. Most of the cases, this directive states that copyright protects the work of its author's life long and 70 years after the author's death. If the author is not known, protection is 70 years from publishing date of the work. When compared to the term provided in Berne convention Term directive give 20 years longer protection period for the works than Berne convention.[109]

The database directive is about how intellectual properties relating to databases are protected. According to database directive, the design of the database is protected with copyright if the design is non-trivial.[110] Copyright does not protect the data which is in the database. On the other, the data is protected with *Sui generis* right for the

[104]Rome Convention for the Protection of Performers, Producers of Phonograms and Broadcasting Organizations 496 UNTS 43.

[105]Justine Pila and Paul Torremans, *European intellectual property law* (Oxford University Press 2016) p. 247.

[106]DIRECTIVE 2009/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 April 2009 on the legal protection of computer programs [2009] OJ L111/16.

[107]Directive 2006/116/EC of the European Parliament and of the Council of 12 December 2006 on the term of protection of copyright and certain related rights [2006] OJ L372/12.

[108]Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases [1996] OJ L77/20.

[109]Directive 2006/116/EC of the European Parliament and of the Council of 12 December 2006 on the term of protection of copyright and certain related rights [2006] OJ L372/12.

[110]Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases [1996] OJ L77/20, art 3.

database.[111] Sui generis right means that the right which protects database data is not similar to the protection of any other object which is protected with intellectual property.

4.1.3 Copyright in Finland

In Finland, copyright is regulated in the Copyright act.[112] Its first section states that a person who has created *"a literary or artistic work shall have copyright therein, whether it be a fictional or descriptive representation in writing or speech, a musical or dramatic work, a cinematographic work, photographic work or other work of fine art, a product of architecture, artistic handicraft, industrial art, or expressed in some other manner and that maps and other descriptive drawings or graphically or three-dimensionally executed works and computer programs shall also be considered literary works"*[113]. This definition is the mostly same definition for copyright which can be found from Berne Convention and computer program directive. The similarity is reasonable because both documents bind Finnish legislators.

Section four of the act is about adaption or conversion of the work. This section states that if someone has adapted copyrighted work or converted it to another form, the adaptor or convertor has copyrighted to this form. However, it is limited to extend in which copyright does not conflict with the copyright of the original work. The limitation means that the adapter has copyright only the part of the work which is his or her creation. On the other hand, if the adaption can seem independent, then the copyright does not depend on original work.[114]

[111]Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases [1996] OJ L77/20, art 7.

[112]Copyright Act, 404/1961.

[113]Copyright Act, 404/1961, s 1.

[114]Copyright Act, 404/1961, s 4.

Section five of the act is about collective work. Collective work is the work which contains part from many other works. In this situation, copyright of the compilation work belongs to the person who has made the compilation work again with the limitation that these rights cannot collide with rights that belong to the copyright holders of the original works.[115]

Copyright distribution of the work, which has multiple authors, has covered section six of the act. According to section six of the copyright act, if it is impossible to determine who has authored which part of the work, then the copyrights belong to each author equally.[116]

4.2 Software copyrights

According to Computer Programs Directive[117], copyright protects computer programs. The directive also states that ideas and principles which are useable in many programs can not be protected. In addition to that computer programs should be its author intellectual creation and original.[118]

Computer Programs Directive also states that the author of the software can be a natural person or group of natural persons. For a group of persons, the directive mentions two different types of work. Suppose persons create work such that it is impossible to determine which person has created which part of the work, then all creators are rightsholders jointly. On the other hand, if parts can be identified, collective work and

[115]Copyright Act, 404/1961, s 5.

[116]Copyright Act, 404/1961, s 6.

[117]DIRECTIVE 2009/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 April 2009 on the legal protection of computer programs [2009] OJ L111/16.

[118]DIRECTIVE 2009/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 April 2009 on the legal protection of computer programs [2009] OJ L111/16.

rights are distributed according to member state legislation. Furthermore, if developers have made the software program as a part of their job, copyrights of the program belong to the employer if nothing else is contracted.[119]

Because copyright protects computer programs, Computer Program directive provides exclusive rights for rightsholder are similar to exclusive rights for another type of artworks which are copyright protectable. According to the directive, rightsholder has exclusive rights to reproduce temporarily or permanently. Rightholder also has the right to translate and adapt the software. Distribution right is also exclusive right of the rightsholder.[120]

Article 6 of the directive set boundaries for reverse engineering. Reverse engineering is the process in which the object code of the software is converted back to the source code. Under article 6, reverse engineering is allowed only to make other software communicate with the software. There are also requirements that user have rightful access to the software, and interoperability cannot be achieved without reverse engineering. Furthermore, only part of the software which is relevant to interoperability can be a target of the reverse engineering process.

Article 8 of the Computer Programs directive contains special measures of the protection. The article states that person who put into circulation a copy of a computer program knowing, or having reason to believe, that it is an infringing copy shall be a target to remedies. Same applies to the person who uses the commercially infringing copy of the software.

[119]DIRECTIVE 2009/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 April 2009 on the legal protection of computer programs [2009] OJ L111/16, art 2.

[120]DIRECTIVE 2009/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 April 2009 on the legal protection of computer programs [2009] OJ L111/16, art 4.

5 Authorship of software

5.1 Problems in open-source authorship

There are many problems in the question about right for remedies in license infringements when it comes to open source licenses. Origin of all problems is in the legal sense challenging form of the open-source software project. Some of the problems are related to the fact that the development of the open-source software projects commonly take place in the global and virtual workspace, where each developer's factual contribution is not always necessarily clear. The others are related to the software's legal status, which is the end product of the project.

First of all, it is challenging to identify who are contributors to a particular block of code. Identifying a contributor is a challenge because development is international, and in many cases, it is not even necessary to use a real name when contributing open-source software project. Then again, one may contribute block of code, and after that, others modify half of this block. After the second modification of the line, there are two contributors for the one cone line, and it is hard to discover who have contributed and what is a contribution. One solution for this problem is to check contributions from the version control but also in this solution have its downsides. Another solution which has the potential to solve the problem of tracking authorships is blockchain, and these solutions are also introduced in the following sections.

Another thing to consider is what are the requirements that software even is copyright protectable. Protectability is an important question because copyright protects only the author's intellectual creations[121] and therefore, not all code is protectable. Some software is so trivial that they do not have copyright protection. Moreover, if the partition of the code which a contributor has contributed is not intellectual enough, then it is not copyright protected.

5.2 Proving authorship with Git

5.2.1 Git version control system

The version control system is software that keeps track of different versions of the developed software. Version control is essential for the software development process because it makes it possible to find the last version of the software where some feature works if the feature is broken during development. Because it is possible to identify the last version where feature works, it is easy to find the reason why the feature has broken just looking what has changed between version where the feature worked and the version where it does not work.[122]

Although there are many version control systems[123], most used version control for open source projects is Git[124] which nearly 87 % of the software developer use.[125] Linus Torvalds initially developed it for the version control of the Linux operating system

[121]DIRECTIVE 2009/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 April 2009 on the legal protection of computer programs [2009] OJ L111/16, art 1.3.

[122]More about version control systems, e.g. Walter F Tichy, "RCS—a system for version control" (1985) 15(7) Software: Practice and Experience 637.

[123]for instance SVN, Bitkeeper, CVS

[124]Git is free and open-source software licensed under GPLv2 and can be downloaded at https://git-scm.com/

[125]"Developer Survey Results 2018" (*Stackoverflow*) ⟨https://insights.stackoverflow.com/survey/2018/#work-_-version-control⟩ accessed 6 June 2020.

in 2005. The initial reason for the development of Git was that Linux project lost its license to use Bitkeeper.[126]

As Git is a commonly used version control system for open source projects, it is a good starting point when searching for ways to prove authorship. One great feature in Git is that it collects data about authors for codebase which versions it control. When Git tracks changes in source code, it is possible that at the same time, it collects data based on which it is possible to identify the author of each line of code.

Git is a decentralized version control system. Decentralized means that there is no authoritative repository, which is a difference in centralized version control systems. Because Git is a decentralized version control system, it uses repositories which are "cloned" from each other and contains full editing history and all other data. These repositories are initially similar. During the time repositories can vary, and they can import and export different blocks of code to and from other repositories. Although Git version control does not by definition contain authoritative repository which contains the newest version about code, in reality, there is in most cases one repository from which all other are cloned.[127]

In imports of the code block, which are called commits, repository imports code blocks from another repository, but in addition to that, it gets metadata about the code block. The labels of the metadata in commit are commit id, author, author date, committer, commit date, and Git commit message. Commit id is a unique id that is used to identify the commit. The author is the person who committed the code block to Git. The date in which code is committed Git is saved to the author data field in metadata.

[126]Junio C Hamano, “GIT–A stupid content tracker” (2006) 1 Proc. Ottawa Linux Sympo 385, p. 385.

[127]Armijn Hemel and Shane Martin Coughlan, “Making Sense Of Git In A Legal Context” (2017) 9 IFOSS L. Rev. 19, p. 20.

Then again, the committer is the person who committed the commit to the repository, and the commit date is the date of that event. Git commit message is a message from the person who committed the change. Most likely, it contains a description of change made to the code. The commit can also contain tags which inform for instance that the commit belongs to the specific version of the software.[128]

5.2.2 Solution with GIT

At first, it seems like using metadata from Git to find authorship all code block is a solution for the question about who has written and which part of the code. Some scholars state that metadata is an efficient way to maintain the value of copyright in the digital environment. Maintaining the value of copyright means that it would be easier to track what rights each party have.[129] On the other hand, to benefit from the metadata, the form of it must be correct. There is no use for the metadata which contains unnecessary information or even worse invalid information about copyright relationships.[130]

Unfortunately, no mechanism provides that metadata from the Git contains correct information about, for example, the author of the code block. The code block may be copied from one project and pasted the other project. In this case, copyright belongs to the original writer of the code block not to the person who copied it although his name is in the metadata. There are no evidence about copying other than that same

[128]Armijn Hemel and Shane Martin Coughlan, "Making Sense Of Git In A Legal Context" (2017) 9 IFOSS L. Rev. 19, p. 24.

[129]Leo Mullins, "Using metadata to support DRM, trading and administration of globally deployed digital products" (2009) 5(2) Journal of Digital Asset Management 75, p. 76.

[130]Armijn Hemel and Shane Martin Coughlan, "Making Sense Of Git In A Legal Context" (2017) 9 IFOSS L. Rev. 19, p. 28.

code is also in the source code of that other project.[131] On the other hand, Git is decentralized version control which provides that it is hard to set incorrect information to the metadata because there are multiple copies about the repository, but it is not impossible.[132]

Another problem is that Git tracks change code line wise. Line wise tracking means that if the developer change one character in the line, Git considers that this developer has written this whole line. Although this is not a typical problem current, someone may rename variables. After committing these changes, he is considered as an author of all lines in which these variables occur. All this because he was the last person who modified these lines. It is easy to see that these modifications do not fulfil the requirements set to the copyright protection for computer programs.

There is one law case in Germany[133] where Git logs are used as a piece of evidence for copyright infringement. *Hellwig v. VMWare Global Inc.* was about Virtual machine which VMWare has developed. The virtual machine is the software that emulated hardware[134]. As a part of VMWare's product, VMWare was developed files vmkernel and vmlinux. Because vmlinux contained source from the Linux operating system which is licensed under GPL license, also vmlinux was licensed under GPL, and its source code

[131]Armijn Hemel and Shane Martin Coughlan, "Making Sense Of Git In A Legal Context" (2017) 9 IFOSS L. Rev. 19, p. 28.

[132]using git push –force command it possible to change also older commit therefor inject incorrect data, but that is quite easy to notice because it changes commit date. See. "git-push documentation" (*Git documentation*) ⟨https://git-scm.com/docs/git-push#Documentation/git-push.txt--f⟩ accessed 6 June 2020.

[133]German case law is covered in that and same other occasions in that thesis because many open source infringement case take place on Germany because of German procedural laws. See. Marcus von Welser, "Opposing the Monetization of Linux: McHardy v. Geniatech & Addressing Copyright Trolling in Germany" (2018) 10 IFOSS L. Rev. 9, p. 11.

[134]See. Gerald J Popek and Robert P Goldberg, "Formal requirements for virtualizable third generation architectures" (1974) 17(7) Communications of the ACM 412.

was published. Then again vmkernel did not contain code from the Linux project, and therefore its source code was not revealed. The legal question in the case was whether vmkernel is derivative work to the GPL licensed Linux code. In its ruling, Hamburg District Court states Hellwig did not express it clear enough which part of the code is his contribution, and that code is intellectual enough to be copyright protectable. On the other hand, Git itself was not questioned as a form of the procedural evidence.[135]

Although there are evident, practical problems, case law states that Git log data is enough evidence about authorship at least in Germany.[136] On the other hand, it requires that plaintiff express clearly which part are similar and which notations in the Git logs support the claim that the defendant has committed copyright infringement.

5.3 Proving authorship with Blockchain

5.3.1 Blockchain technology

Blockchain technology is another solution for metadata in which incorrect data is hard to inject. The validity of the metadata is important because metadata is useless if it is possible to any change it as they wish, mainly when the metadata shall provide the information about the author of the code block. The most final solution for that purpose is the blockchain application.

According to the definition provided by Founder and Executive Chairman of the World Economic Forum Klaus Schwab, blockchain is *"[i]n essence, the blockchain is a shared, programmable, cryptographically secure and therefore trusted ledger which no single user*

[135]Hellwig v. VMWare Global Inc., File no: 310 0 89/15, Hamburg District Court (Jul. 8, 2016)

[136]Armijn Hemel and Shane Martin Coughlan, "Making Sense Of Git In A Legal Context" (2017) 9 IFOSS L. Rev. 19, p. 19.

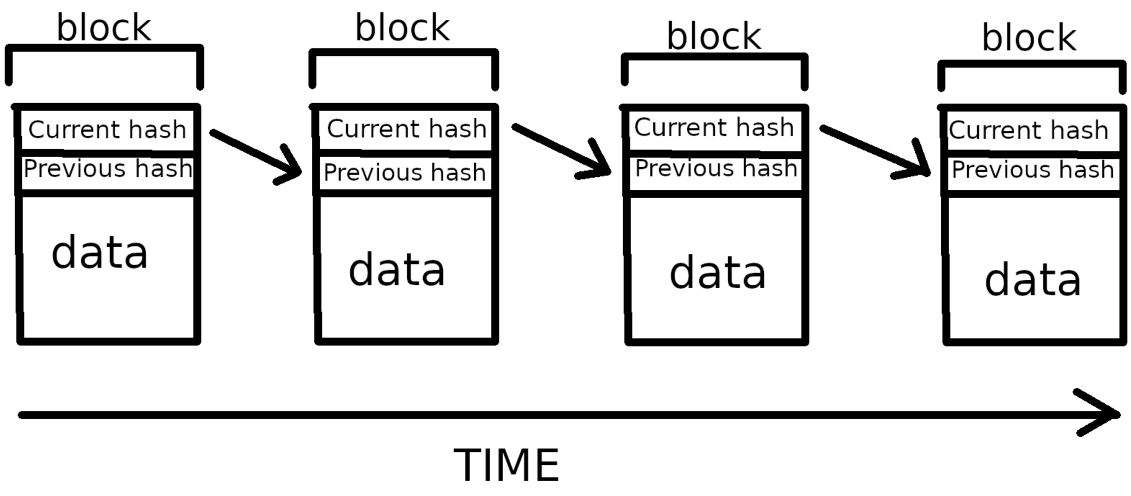


Figure 5.1: Illustrative image about blockchain

controls and which can be inspected by anyone".[137] Blockchain protocol is designed such that it is tough to make hostile changes to the data stored to the blockchain.[138]

The basic idea in blockchain as a data structure is that it contains blocks which have hash part and data part (see. figure 5.1). Data part has data which belongs to the block, for instance, who owns the copyright of the specific artwork. Then hash part have cryptographic hash from both previous and current block.[139]

Hashes are calculated with cryptographic function, which input is the data which is saved to the current block and also hash from the previous block. The output of the function is hash which is a series of hexadecimal numbers. A hash of the previous block

[137]Klaus Schwab, *The fourth industrial revolution* (Currency 2017) p. 19.

[138]technically changes are possible with so-called "51% attack", but that requires that attacker have at least 50% of all computation power in the blockchain. The situation is theoretically possible but with most of the blockchain nearly impossible in practice. see. Paul Vigna and Michael J Casey, *The age of cryptocurrency: how bitcoin and the blockchain are challenging the global economic order* (Macmillan 2016).

[139]Michael Crosby and others, “Blockchain technology: Beyond bitcoin” (2016) 2(6-10) Applied Innovation 71, p. 11.

is one input when computing a hash for the next block, anyone who wants to change in the earlier block in the blockchain has to compute the new hash for that block and also all the following block.[140]

Another factor which validates data storage in the blockchain is the consensus requiring and decentralized nature of the blockchain technology. Decentralized nature means that there are multiple copies about blockchain and they are all equally valid. These places where blockchain copies are stored are called nodes. When a new block is added to the blockchain, there is a typically same computationally challenging problem which must be solved before the block can be added to the chain. For instance, in the Bitcoin blockchain, all block hashes must start which specific number of zeros and for that reason block contains a nonce, which is number selected such that requirement is fulfilled. When one node find the solution, others check that it is correct and after that, all nodes adds the new block to their versions of the blockchain.[141]

5.3.2 Solving authorship with blockchain

Technically it is possible to write right owner information to the blockchain, and that way solve who owns the copyright for the specific work. Storing copyright information can be done with "Trusted Timestamping". That is encoded sequence of characters which describe for instance date of the creation for the artist work.[142]

It is also possible to update the information when someone transfers copyright to

[140]Michael Crosby and others, “Blockchain technology: Beyond bitcoin” (2016) 2(6-10) Applied Innovation 71, p. 11.

[141]Michael Crosby and others, “Blockchain technology: Beyond bitcoin” (2016) 2(6-10) Applied Innovation 71, p. 11.

[142]Alexander Savelyev, “Copyright in the blockchain era: Promises and challenges” (2018) 34(3) Computer law & security review 550, p. 553.

another party. Transfer of the rights can be stated with another timestamp with present time when rights are moved. That way anyone can recognize when rights are moved and also who own right currently.[143]

What is problematic for the blockchain is that it requires an increasingly high amount of the data storage and at least in some implementation computation energy. At the time of the writing estimated energy usage of the bitcoin blockchain, which is most used blockchain is 61 terawatt hour per year. The amount of energy is nearly the same as the amount of energy which is used in Kuwait in one year.[144] When it comes to storage usage, it is clear that over time needed storage increase as blockchain technology to require that multiple nodes stores all blocks.[145]

Another storage-related issue is whether copyright-protected work should be stored in or outside of the blockchain. Both solutions have problems. If works are stored inside blockchain that increase significantly required storage space and in fact, is in conflict with copyright legislation. There is the conflict because parties who maintain nodes for blockchain can be considered as an online intermediary. Therefore according to E-commercial directive, they have to remove copyrighted material if the copyright holder requires it. The requirement for removing data is problematic in the blockchain context because it is impossible to remove data from the blockchain. On the other hand, if works are stored outside the blockchain, the problem is that there should be a mechanism which ensures that data stored inside the blockchain and data stored outside the blockchain is synchronized. This solution also requires a trustworthy party

[143]Alexander Savelyev, "Copyright in the blockchain era: Promises and challenges" (2018) 34(3) Computer law & security review 550, p. 554.

[144]"Bitcoin Energy Consumption Index" (*Digiconomist*) ⟨https://digiconomist.net/bitcoin-energy-consumption⟩ accessed 28 July 2020.

[145]Alexander Savelyev, "Copyright in the blockchain era: Promises and challenges" (2018) 34(3) Computer law & security review 550, p. 551.

who stores the data which is outside the blockchain.[146]

5.4 Intellectual creation requirement for softwares

EU legislation states that the work should be its author's own intellectual creation to get copyright protection. That requirement is initially established in *Infopaq*[147] case, and for the software, it is also stated in the Computer Program directive article 1.3. In *Infopaq* question was whether it infringed copyrights of the newspapers when Infopaq provided eleven words long extracts about their articles. In its decision, ECJ states that eleven-word long extraction is not long enough to infringe rightsholder's exclusive right to reproduce. In addition to that, the court stated that copyright protection requires that work is the author's own intellectual creation. To end up that decision, the court interpreted Berne convention such that it contains intellectual creation requirement in article 2.5 and 2.8[148].

In general threshold for computer software, copyright has not been high. The basic idea is that copyright protects expression in the source code of the software if there are many ways to express the same thing, but if there is only one way to write one code then copyright does not protect it because there is no creativity.[149] On the other hand, there are certain clear limits about what kind of subject can be protected with copyright. These limits are set in ECJ cases *SAS v. the WPL*[150] and *BSA*[151].

[146]Alexander Savelyev, "Copyright in the blockchain era: Promises and challenges" (2018) 34(3) Computer law & security review 550, p. 556.

[147]Case C-5/08 *Infopaq International A/S v Danske Dagblades Forening* [2009] ECR.

[148]Case C-5/08 *Infopaq International A/S v Danske Dagblades Forening* [2009] ECR.

[149]Mikko Välimäki, *Oikeudet tietokoneohjelmistoihin* (Talentum 2009) 18.

[150]Case 406/10 *SAS Institute Inc v World Programming Ltd,* [2012] ECR.

[151]Case 393/09 *Bezpečnostní softwarová asociace - Svaz softwarové ochrany v Ministerstvo kultury* [2009] ECR.

Especially the doctrine that copyright protects the expression of the source rather than functionality is established *SAS v. the WPL*[152] case. Background of the case was thas SAS Institute was developed statistical programming language and development tools for that programming language. After that, World Programming Ltd (WPL) developed software that could compile the SAS programming language.

The legal question in *SAS v. the WPL* was whether WPL was violated copyrights owned by SAS Institute. In this case, ECJ ruled that *"the functionality of a computer program nor the programming language and the format of data files used in a computer program in order to exploit certain of its functions constitute a form of expression of that program and, as such, are not protected by copyright in computer programs for that directive."*[153]

In *BSA* case legal question was whether the graphical interface is copyright protectable as a part of the computer program. In its decision, ECJ states that copyright does not protect the graphical interface of the computer software according to the Computer program directive. On the other hand, the graphical interface may get copyright protection based on Infosoc directive if it is creator's own intellectual creation.[154]

[152]Case 406/10 *SAS Institute Inc v World Programming Ltd,* [2012] ECR.

[153]Case 406/10 *SAS Institute Inc v World Programming Ltd,* [2012] ECR.

[154]Case 393/09 *Bezpečnostní softwarová asociace - Svaz softwarové ochrany v Ministerstvo kultury* [2009] ECR.

6 Enforcebility of Open source licenses

One problem with open source licenses is whether one right owner can raise a claim on behalf of the other. This problem is highly related to the problem of the project's intellectual property status; in other words, whether the software is considered as jointly authored work of collective work. The solution is easy if the software is jointly authored work because, in this situation, every rightsholder can raise a claim. The problem is more challenging if the software is considered as the collective work. In this situation, the contributor can raise a claim on behalf of itself but not others. A possible question is if it is possible to authorize another contributor to run a claim also on behalf of the others.

6.1 Legal status of Open source licenses

It was long uncertain if open-source licenses are enforceable because there was no case law about it. One reason for the lack of case law was that nearly all potential cases were settled before they go to court. Furthermore, the reason for the settlement was at least partly the fact that without caselaw, there was no certainty about the outcome of the case, and neither of the parties was ready to take the risk. The primary reason for the uncertain outcome was that the legal status of open source licenses was unclear. There was no evidence that they are legally binding and if they are whether they are licenses or contracts.[155]

It is vital to mention that in that uncertain situation licensees could only lose in the court. The reason was that even if the court had ruled that open source license is invalid, the copyright legislation would still apply and the outcome would be same or worse when compared to the situation in which open source license is valid.[156]

Another reason for the lack of cases was in the open-source developer community. According to Ellickson Robert, if community have a more efficient way to settle disputes in the community, it will use it.[157] That was the case in the open-source community. The community used many informal ways when sought license compliance. These ways can be for instance sending email to license infringer or write a post to the online forum that someone does not act license compliant way.[158]

The problem for enforceability of open source licenses has also been that according to FSF, the GPL license is not a contract but a unilateral license. This interpretation aimed to prevent problems that might occur with contract formation doctrine in the US in the 1990s. Although doctrine has changed, FSF has kept its position.[159]

[155]Heather J Meeker, “Open Source and the Age of Enforcement” (2012) 4 Hastings Sci. & Tech. LJ 267, p. 268.

[156]Ville Oksanen and Mikko Välimäki, “Free software and copyright enforcement: A tool for global copyright policy?” (2006) 18(4) Knowledge, Technology & Policy 101, p. 104.

[157]Ellickson Robert, *Order without Law–How Neighbors Settle Disputes* (Harvard University Press 1991) p. 280.

[158]Ville Oksanen and Mikko Välimäki, “Free software and copyright enforcement: A tool for global copyright policy?” (2006) 18(4) Knowledge, Technology & Policy 101; Siobhán O’Mahony, “Guarding the commons: how community managed software projects protect their work” (2003) 32(7) Research policy 1179.

[159]Heather J Meeker, “Open Source and the Age of Enforcement” (2012) 4 Hastings Sci. & Tech. LJ 267.

In the US, it was *Jacobson v. Katzer*[160] case, which clarified the legal status of the open-source licenses. In the case, the facts are that both parties developed software for model railroads controlling. "Java Model Railroad Interface(JMRI)" was developed by Robert Jacobsen under Artistic license when Matthew Katzer developed his program under a proprietary license. Initially, Katzer sued Jacobsen with the claim that Jacobsen was breached the software patent owned by Katzer. When this case was in the process, it transpired that Katzer was used code from Jacobsen's JMRI project against license provisions. The case stated that open-source licenses are not unenforceable because of their form and on the other hand, there is no conflict with contract formation rules.[161]

In Finland, open-source licenses are interpreted as standard licenses and, therefore, as binding as other standard licenses used in the software market. That is why all legislation and case laws relating to copyright licenses are also applicable to open source licenses.[162] As licenses are in general interpreted to contract, then open-source licenses are also contracts in Finnish legislation.[163]

6.2 Joint authored, adaptive or collective work

It is essential to discover whether open-source software project is considered as jointly authored, adaptive, or collective work in the copyright sense. Discovering the copyright type of work is important because enforcing rights is different in these cases. In jointly authored work, any one of the authors can raise a claim.[164] On the other hand in

[160] *Jacobsen v Katzer* 609 F Supp 2d 925 (ND Cal 2009).

[161] Heather J Meeker, "Open Source and the Age of Enforcement" (2012) 4 Hastings Sci. & Tech. LJ 267, p. 276.

[162] Mikko Välimäki, "Avoimen lähdekoodin ohjelmistolisensseistä" (2002) 5 Defensor Legis, p. 854.

[163] Kristiina Harenko, Valtteri Niiranen, and Pekka Tarkela, *Tekijänoikeus. Kommentaari ja käsikirja* (Talentum 2006) 488,494.

[164] Copyright Act, 404/1961, s 6.

collective work and adaptive work, creator of the collective or adaptive work can raise a claim but only such way that it does not collide with the copyrights of the original work.[165] Difference of these type of work is defined based on Finnish copyright law in section 4.1.3 and illustrated in figure 6.1. All of these interpretations are, at least in theory, possible for open source projects.

In many cases, it is possible to track who has coded which part of the software. Therefore source code can be seen as a group of authorial work collected to the same work. That is why collective work is a possible category for the source code of the open-source project.

On the other hand, the software development process can be seen as a process where each developer, in turn, adapt the current version and makes it better that way. That way, contributors not only add new increments to work but also further develop old parts. From this perspective, adaptive work is a reasonable interpretation.

Then again, all contributions are contextually related to each other such that it is not possible to separate parts from each other. From that perspective, it is the same to divide code blocks based on their author than divide sentences from the novel based on who has written them. With that reasoning, it is possible to end up the conclusion that open source projects are jointly authored works.

In the following subsection, this problem is discovered from both software engineering and legal perspective. These perspectives are different because software engineers see software as an end product of the development process when lawyers see software as authorial work protected with copyright. It is essential to understand both perspectives because the law system does not live in a vacuum but is part of the society.

[165]Copyright Act, 404/1961, ss 4,5.

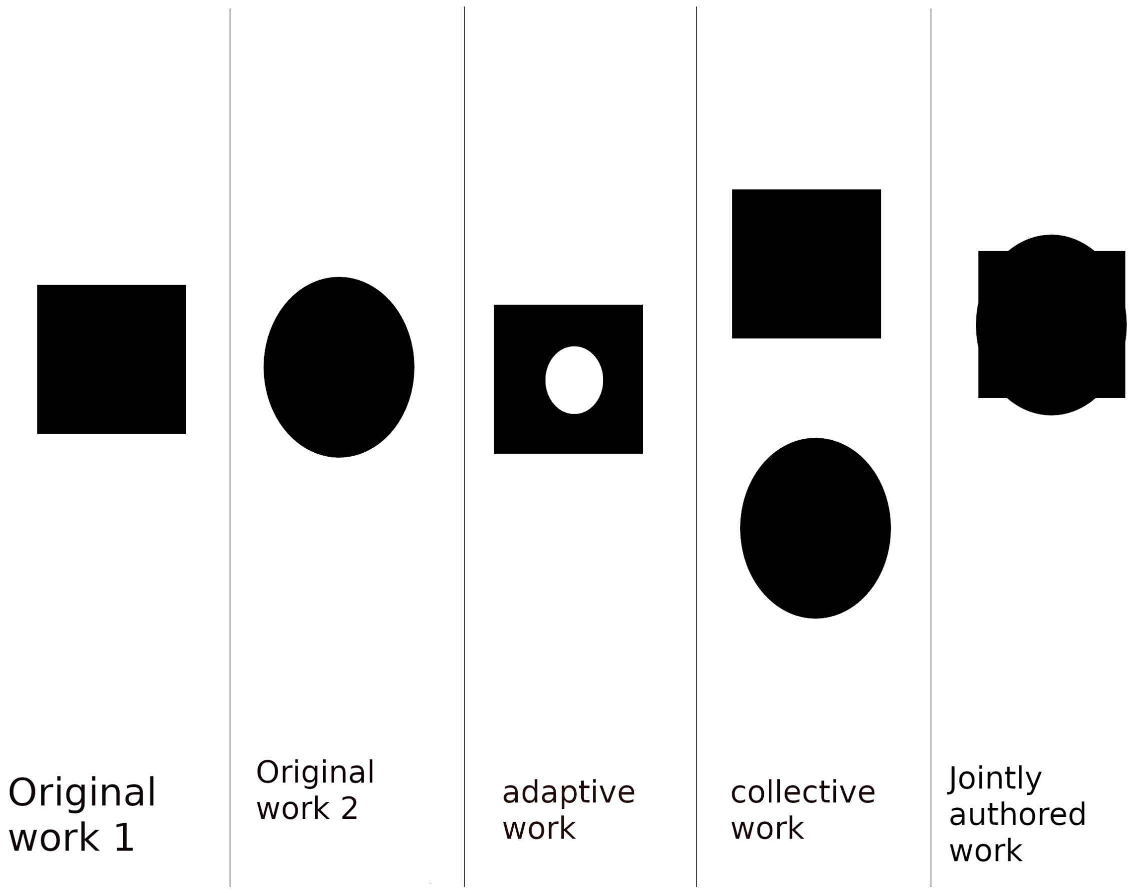


Figure 6.1: Illustrative image about different type of copyrighted works with multiple authors

6.2.1 Technical perspective

From the software development perspective open source project is one software entity which contains many separate code files. These files then contains rows of source code. Every line can have separate author but they are still part of the code in functional sense.

From this perspective, the project is one functional set. All code files are connected, and the software works if they are all present. If any line of code is removed, the program will not work the same way.

When contributor modifies code, it either adds, removes or changes the line of code. When the contribution is ready, the contributor has to check that the software still works with contributor contribution. That way also lines which the contributor does not modify influence the way contributor contributes to the code.

The facts mentioned above support the interpretation that the open-source software is either jointly authored or adaptive work from the software development perspective. The motivation for this opinion is that the software is one entity, and this limits off collective work because it has multiple noticeably different sections.

The fact is also that new version of the end product of the project is not released after each modification to the code, but multiple modifications are published in one release.[166] Thus, it is reasonable to conclude that software engineers see open source projects in most cases as jointly authored work. The exception can be the first versions of the forked projects because they are adaptions from the original works.

One conflict between software development and the copyright is also how to system design should be recognized in the copyright sense. For instance, software architectures and other design materials are not commonly protected as a part of the computer software. On the other hand, these materials can be protected with copyright if their own artist value is considerable enough, but that does not prevent others from implementing architectures which are presented in the material.[167]

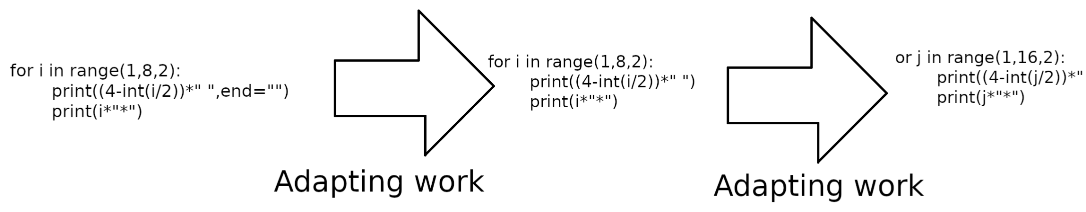
As module design is in many software projects essential part which reflects robust

[166]Linus Nyman and others, "Understanding Code Forking in Open Source Software: An examination of code forking, its effect on open source software, and how it is viewed and practiced by developers" [2015] , p. 26.

[167]Mikko Välimäki, *Oikeudet tietokoneohjelmistoihin* (Talentum 2009) p. 21.

to choices will are made when the software is coded[168]. It is one question whether copyright belongs to the programmer if the module design leaves narrow space for the programmers own artistic choices. In the end, that does not matter if module designer and programmer work in the same company because copyright belongs to the company but in some open source project that is not the case.

6.2.2 Legal perspective



```
for i in range(1,8,2):
    print((4-int(i/2))*" ",end="")
    print(i*"*")
```

```
for i in range(1,8,2):
    print((4-int(i/2))*" ")
    print(i*"*")
```

```
or j in range(1,16,2):
    print((4-int(j/2))*" ")
    print(j*"*")
```

Figure 6.2: Picture about fictional opensource software development process where the aim is making a program which prints an increasing number of asterisks to every other line

When software engineering sees code as a product, the legal view is different. From the legal perspective, computer programs are protected as literate work.[169] This form influences the answer because the literate form only considers source code, not the functionalities. Not recognizing the value of functionality is the opposite view than what is in software engineering because software engineering is more interested in function-

[168]Bernd Bruegge and Allen H Dutoit, *Object–Oriented Software Engineering. Using UML, Patterns, and Java* (Pearson 2014) p. 217.

[169]DIRECTIVE 2009/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 April 2009 on the legal protection of computer programs [2009] OJ L111/16.

alities of the code than how it is written.[170]

From *SAS v. the WPL*[171] case, it can be interpreted that copyright-protection does not require that software works. This interpretation can be made because the functionality is not protected, but the expression of source code is. With the fact that the function of the code is measurable only when the source code is compiled to object code and program executed in the computer, it is not possible to derive requirements about protectability when the software is protected as authorial work.

Besides, contributing open source project means in the legal sense that contributor license his code to the rights owner of the project with the same license as the project use.[172] From this perspective, open-source software seems to be a collection of code blocks from different authors, and that is collective work.

It is also reasonable to notice that not all contribution is protected with copyright. There are still some requirements for the code's creativity than are for the "normal" computer program. Each contribution should be "its author intellectual creation"[173] to be protected with copyright.

In German case law contributions to the software is considered to be jointly authored only if the contribution is from the contributor who has provided to the initial version of the software[174]. Another wise contribution is collective and to get copyright protection for the contribution it is required that contribution independently fulfil requirements

[170]That does not mean that formation of the code is meaningless in software engineering. Code must be readable, but it is more important because it does what it aims to do

[171]Case 406/10 *SAS Institute Inc v World Programming Ltd,* [2012] ECR.

[172]In same projects there is also a Contributors license agreement, which states the legal relationship between the contributor and the rightsholder of the project preciously.

[173]DIRECTIVE 2009/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 April 2009 on the legal protection of computer programs [2009] OJ L111/16.

[174]German Federal Supreme Court, Judgment of July 14, 1993, File No.: I ZR 47/91

for the copyright protection[175]. In the infringement situation, it is also necessary that contributors show what their contribution is. The requirement is that contributors show exactly which lines of the code are their contribution and which lines are intelligent enough to be protected with copyright. It is also essential to show in which code lines defendant have used.[176]

[175]German Federal Supreme Court, Judgment of Mar. 3, 2005, File No.: I ZR 111/02.; Fash 2000

[176]Marcus von Welser, "Opposing the Monetization of Linux: McHardy v. Geniatech & Addressing Copyright Trolling in Germany" (2018) 10 IFOSS L. Rev. 9.

7 Open source infringement litigation

7.1 Raising a claim on behalf of other contributors

According to decision KKO 2004:18 from the Finnish Supreme court, there is no way to mitigate the right to raise a claim to another person. In that case, person A claims that real estate in which person A owned part should modify its articles of association more reasonable. During the court process in the court of appeal, person A sold his part of the real estate but continued his process in the court with the new owner's acceptance. In this situation, the Supreme court ruled that although person A has no right to continue the process anymore because person A does not have ownership, which was a motive for the process. The court also ruled that it does not change the situation that the new owner accepted that person A continues to process.

Also, the Finnish market court has made judgment which is based on decision KKO 2004:18 and is related to copyright. In the case of the market court MAO:285/19 critical legal question was whether right performance organization Teosto could raise a claim on behalf of the right owners which right it protects. Reasoning its decision with the decision KKO 2004:18, The market court states that although Teosto had contracts with its member, it cannot present them in the court. The contracts did not affect to outcome because the right to raise a claim cannot move even contractually to another person.

There is still one way to make it possible to file cases on behalf of all right owners of the end product of open source projects. The class action is "a legal action organized by a group of people who all have the same legal problem"[177]

Section 2 of the Finnish Act on Class Actions set three conditions for the class action.[178] The first requirement is that multiple persons have claims against the same defendant based on the same or similar circumstances. This requirement is fulfilled in open source license infringement because every copyright owner, whose right infringer has violated, has claims against the infringer. All claims are based on the same action of the infringer.

The second requirement is that the hearing of the case as a class action is expedient given the size of the class, the subject-matter of the claims presented in it and the proof offered in it.[179] Clearly, from the perspective of the proof, it is convenient to use class action as an offered proof is the same for all contributors cases. Also, the presented claims are similar to all contributors. On the other hand, size is the requirement that some open-source projects fail because they only have few contributors. On the other hand, there are also open-source projects such as Linux kernel which have over 1500 contributors.[180]

The third requirement is that the class has been defined with adequate precision.[181] This requirement is easily fulfilled with open source projects because the group of claimants is preciously the same as a group of contributors.

Unfortunately, Finnish legislation does not allow the use of class action in the context of open source license infringement because Act on class action only permits the use

[177]"Class Action" (*Cambridge Dictionary*) ⟨https://dictionary.cambridge.org/dictionary/english/class-action⟩ accessed 30 May 2020.

[178]Act on Class Actions, 444/2007, s 3.

[179]Act on Class Actions, 444/2007, s 3.2.

[180]"2017 State of Linux Kernel Development" (*Linux foundation*) ⟨https://www.linuxfoundation.org/2017-linux-kernel-report-landing-page/⟩ accessed 31 May 2020.

[181]Act on Class Actions, 444/2007, s 3.3.

of class action in consumer cases. Only customer ombudsman can run class action in the court. That is why open source license infringement cases can not be class action as contributors are not in the position of customer relating to the license infringer. Finnish legislator made that limitation to prevent partisan cases where a class action is started just to harm defendant company[182].

Because open-source license infringements are not suitable for class actions, it means that in an open-source software project, the contributor can raise claims only on behalf of himself or herself and on behalf of a company. In other words, it is not possible that the contributor raises claims in the court on behalf of all contributors to the project.

7.2 Possible Claims

7.2.1 EU Legislation

In general, enforcing methods which are available for intellectual property disputes in the EU are regulated in Enforcement directive.[183] According to its first article "[T]his Directive concerns the measures, procedures and remedies necessary to ensure the enforcement of intellectual property rights".[184]

In the case of open-source licenses, it is not that clear if enforcement directive applies. the reason for that is that the directive does not explicitly state that it applies

[182]Mikko Välimäki, "Introducing Class Actions in Finland: An Example of Law-making Without Economic Analysis" in *The Law and Economics of Class Actions in Europe* (Edward Elgar Publishing 2012) p. 333.

[183]Corrigendum to Directive 2004/48/EC of the European Parliament and of the Council of 29 April 2004 on the enforcement of intellectual property rights [2004] OJ L195/16.

[184]Corrigendum to Directive 2004/48/EC of the European Parliament and of the Council of 29 April 2004 on the enforcement of intellectual property rights [2004] OJ L195/16, art 1.

enforcement of licenses but the intellectual property itself.[185] Another reason is that sometimes open-source licenses are interpreted as contracts, and in this case, it is a national contract law that applies not the intellectual property law.

The solution to this problem can be found from case law. The ECJ case *IT Development v. Free Mobile*[186] was about whether the Enforcement directive applies to license infringement when the licensee was against the license modified the licensed software. In this case, ECJ ruled that enforcement directive was applicable. The court reason the judgment with the fact that according to computer directive, the software is protected with copyright with is an intellectual property right as it is meant in enforcement directive article 1. In addition to that, computer directive states that modifying software is one type of copyright infringement related to the software.

It is not clear whether *IT Development v. Free Mobile* states that the Enforcement directive also applies to the open-source directive. The case was fundamentally about modifying software against license conditions. Modifying software is not possible infringement in open source licenses because they, by definition, allow the licensee to modify source code and do derivative works. On the other hand, there is no reason why the Enforcement directive does not apply for Open source licenses because, as mention above in section 6.1, open-source licenses are as enforceable as other software licenses.

As enforcement directive applies to the open-source software licenses, all enforcement methods listed in enforcement directive can be claimed in case of open source software license infringement. Because of the principle of effectiveness in EU which states that member states have to interpret their procedural and remedy legislation such that it

[185]Corrigendum to Directive 2004/48/EC of the European Parliament and of the Council of 29 April 2004 on the enforcement of intellectual property rights [2004] OJ L195/16, art 1.

[186]Case 666/18 *IT Development SAS v Free Mobile SAS* [2019] ECR.

is not in conflict with the EU legislation[187], enforcement directive is applicable in the member states even when they are not implemented it.

According to enforcement directive claiming damages and injunction is possible. The injunction means that the court rules the infringer to stop infringing the right owner's rights. In the context of open source license, the claimant can require that defendant publish source code if this is how the defendant infringe open source license.[188]

7.2.2 National Legislation

When it comes to damages and open source software, it is essential to notice that there are no punitive damages in Finland, unlike in the US. Non-punitive damages mean that the maximum amount of damages is the cost of direct damages which can be seen suffered from actions of potential damages payer.[189] Limitation to direct damages suggests that amount of damages can not be high in open-source cases because a breach of open source license does not directly damage anyone financially. On the other hand according to enforcement directive amount of the damages can also be the amount of money which the infringer save with the copyright infringement.[190] some company may have released their software under open source licenses, which prevents commercial use of the code. In this situation, direct damage is possible if a competitor starts to use the code commercially, and that way breaches the license terms.

Reasonable compensation is also possible according to Finnish Copyright act § 57.2.

[187]Tuomas Ojanen, *EU-oikeuden perusteita* (Edita Publishing Oy 2016) p. 99.

[188]Jennifer Buchanan O'Neill and Christopher J Gaspar, "What Can Decisions by European Courts Teach Us About the Future of Open-Source Litigation in the United States" (2010) 38 AIPLA QJ 437.

[189]Tort Liability Act, 412/1974.

[190]Corrigendum to Directive 2004/48/EC of the European Parliament and of the Council of 29 April 2004 on the enforcement of intellectual property rights [2004] OJ L195/16, art 13.

Reasonable compensation does not require that copyright infringement was intentional, as is the case with the damages. The amount of reasonable compensation has not been as precise for computer software as it is for other copyright-protected work of art. For instance, for music, reasonable compensation has commonly been license fee of the Teosto or Gramex, which are Finnish Central copyright organization for recorded and live music. Because there is no central organization for the computer programs situation is not that clear but according to Finnish Supreme Court subjects which influence compensation is at least standard license fee of the software[191], whether the software is used commercially[192] and whether the software is used the way it is designed to use[193]. The aim in the reasonable compensation is to make it unbeneficial to infringe copyright.[194]

There is also a theoretical possibility of getting contractual compensation when the license is breached because software licenses are considered a contract. On the other hand, this requires that compensation is specified in the license. As none of the popular licenses[195] contains provision about contract fines, it is saved to say that this is not common claim. There are also empirical research which states that damages are not commonly used remedy.[196]

[191]KKO 1998:91.

[192]KKO 1989:151.

[193]KKO 1999:151.

[194]Mikko Välimäki, *Oikeudet tietokoneohjelmistoihin* (Talentum 2009) p. 70.

[195]In this context, popular licenses are what is listed in "Popular Licenses" (*Open Source Initiative*) ⟨https://opensource.org/licenses/⟩ accessed 30 May 2020. At the time of writing these licenses were Apache License 2.0, BSD 3-Clause "New" or "Revised" license, BSD 2-Clause "Simplified" or "FreeBSD" license, GNU General Public License (GPL), GNU Library or "Lesser" General Public License (LGPL), MIT license, Mozilla Public License 2.0, Common Development and Distribution License and Eclipse Public License version 2.0.

[196]Siobhán O'Mahony, "Guarding the commons: how community managed software projects protect their work" (2003) 32(7) Research policy 1179.

The injunction is a more used remedy in open source license enforcement. There are many cases from Germany and France where open source licenser only claims license breacher to stop breaching license and expenses related to court process but not damages.[197]

7.3 Effect of license type

As mentioned in section 2.3, there are multiple different types of open-source license, both based on their functionalities and also based on their historical origin. In many licenses, there is the only provision which is related to the right to use and modify the software and no provision about litigation. As mentioned earlier, none of the most used licenses contains provision about the remedies in the license infringement situation. On the other hand, some licenses contain a clause which is related to litigation.

There is the choice of law and jurisdiction provisions in some open-source licenses.[198] Especially corporate licenses contains these provisions. Most of the choice of law provision in the open-source licenses state that the licenses are governed by the law of some state of US, but that is not the only used provision type. For instance, the Nokia Open source license is governed by Finnish law and dispute are settled by a single arbitrator appointed by the Central Chamber of Commerce of Finland. There are also licenses which state that jurisdiction is based where the defendant is mainly located. The same

[197] Jennifer Buchanan O'Neill and Christopher J Gaspar, "What Can Decisions by European Courts Teach Us About the Future of Open-Source Litigation in the United States" (2010) 38 AIPLA QJ 437 See german cases *Welte v Sitecom* 21 O 6123/04 and *Welte v D-Link* 2-6 O 224/06 and France case *AFPA v EDU4* 04/24298

[198] at least Mozilla Public license, Apple Public Source License and Python License contain that kind of provisions

provision is also used for the choice of law[199].

[199]See, for instance "Mozilla Public License 2.0" (*Open Source Initiative*) ⟨https://opensource.org/licenses/MPL-2.0⟩ accessed 9 August 2020.

8 Conclusion

In the end, it seems that the right to raise claims should be all contributors to the software projects whose contribution itself is copyright protectable. That means not all contributors have that right because some contributions may be generic or else way not enough to be considered as a protectable copyright work. That seems somehow logical result because, for instance, not all writing is protected with copyright. The same of the works are not intellectual enough to be protected.

These contributors are filtered out because open source software is initially jointly authored work which turns to be collective work overtime. In collective work, original rightsholder has the copyright to whole software, but this copyright should be used such that other copyright holders rights are not diminished.[200] That is why contributors whose contribution is not the author's intellectual work[201] cannot raise claim as they do not have any copyrights. This fact is undeniable as a person who does not have copyright obviously can not claim that someone has infringed his copyright. This mention is here in the first place to point out that contributing open source project does not automatically generate copyright to the contributor.

On the other hand, the original copyright holder can raise claims according to these parts of the source code because this does not damage the right holder's right. This kind of activity is possible because according to copyright act[202] the creator of the collective work have the copyright to control work but only such that these right do

[200]Copyright Act, 404/1961, s 5.

[201]requirement for copyright in DIRECTIVE 2009/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 April 2009 on the legal protection of computer programs [2009] OJ L111/16

[202]Copyright Act, 404/1961, s 5.

not infringe right which belongs to rightful owners of those works which are part of the work. As it is impossible to infringe rights that do not exist, the creator of the collective work can raise a claim.

It is interesting to find where there is a line between adaptive work and jointly authored work. At the beginning of the creative process, it is clear that jointly authored work is established in a process that continuously creates adaptive works from previous versions of the work. The joint authors make these versions of the work. That leads to whether activity where certain groups of people make methodically adaptive works from each other works, which are connected to the same project, can create jointly authored work. Because if this is the case, then the open-source projects are jointly authored, and all contributors can raise a claim.

In some open-source software projects, it is challenging to identify who are right owners because of the massive amount of contributors who have contributed the project over time. It is also considered that the copyright does not necessarily belong to the software developer who contributed. The developer may have contributed as a part of his or her work. In that case, copyright belongs to the employer. In that case, besides, to identify contributor also his or her employer is vital to know.

Because finding the right owners is essential in order to prevent license infringements, this thesis has introduced two methods to identify owners of the code. One is to use version control logs to find authors, and the other is to use blockchain to save contributions. The version control method had its benefits because it does not need any additional steps to the development process. Therefore it does not increase the need for data storage from the current state. On the other hand, it requires work to fetch information from the logs to show which contribution of the developer such that it is sufficient enough to be a solid piece of evidence in the court.

When it comes to using blockchain to identify copyright relations in the open-source software project, the benefit is that the system can be designed such that all copyright sense relevant data is collected. Therefore using the data as a piece of evidence in court is easier. The downside is that there are technical difficulties to comply with copyright law. The blockchain also requires more computation power and data storage than version control. The blockchain system is also a new additional system which should be added to the software development process.

In some cases, the problems with ownership of the open-source code are solved, transferring contributors copyrights to the organization which controls the project. That way, the organization has all rights to the source code of the open-source project, and it can start legal actions against license infringers. For instance, Free Software Foundation requires that copyright of every non-trivial change in projects it maintains should transfer to the Free Software Foundation.[203]

In the end, it seems that all research questions covered in this thesis have only theoretical relevance. The situation is that, in most cases, the injunction is the remedy which claimant requires. That is why there is no practical need to determine how all right owners can raise a claim, let alone whether some right owners can raise a claim on behalf of others. The reason for that when one right owner makes injunction claim and the court rule defendant to stop breaching this right owner's rights consequence is that defendant stop breaching any right owner's rights.

The nature of the open-source license is such that all right owners of code in the open-source project give the same rights to the licensee and, on the other hand, set the same obligations. Therefore even though only one right owner require that term of the open-source license are followed with source code that the right owner owns, after the

[203]"Copyright assignment at the FSF" (*Free Software Foundation*) ⟨https://www.fsf.org/bulletin/2014/spring/copyright-assignment-at-the-fsf⟩ accessed 31 May 2020.

defendant has changed its action to comply with the license. After that defendant's action is not only in compliance with terms set by the right owner who sued the defendant but with all other right owners in the open-source project too. Therefore only thing which is needed is that somebody who owns some copyrights to the project starts the legal process. That means that not every contributor can do it because as mentioned above, there may be contributors to the project which does not have any copyrighted contribution in the project.