Master's thesis

Master's Programme in Computer Science

Public copyright licenses in Software Engineering: A Multivocal Literature Review

Akira Taguchi

May 28, 2024

Faculty of Science

University of Helsinki

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki,Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Faculty of Science	Master's Programme in Computer Science

Tekijä — Författare — Author

Akira Taguchi

Työn nimi — Arbetets titel — Title

Public copyright licenses in Software Engineering: A Multivocal Literature Review

Ohjaajat — Handledare — Supervisors

Prof. Tomi Männistö

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	May 28, 2024	22 pages, 2 appendix pages

Tiivistelmä — Referat — Abstract

Context: Public copyright licenses (PCL) are central to the distribution of works in software engineering. For example in open source there must be an appropriate PCL attached to the source code in order for open-source software to be freely available for possible modification and redistribution. Understanding PCLs can be difficult. This could stem from the legal nature of the license texts and the large number of already-existing PCLs. As a result some actions made within the boundaries of the PCLs may come as a surprise to the public.

Objective: The primary goal of this research is to conduct a multivocal literature review of the current state of PCLs in software engineering, the evaluation of the them and the evidence level of the research. The research aims to provide a novel perspective on relevant licenses and to extract key findings through a rigorous literature review process. This study has two main viewpoints: to provide rigorous research on PCLs to the academic field and to provide insights to the professional field of software engineering on PCLs. The grand goal of this thesis is to raise awareness of the importance of PCLs so that more licensers would make the correct choices based on their situations and needs in a mindful way.

Method: The search strategy examined 6666 sources, found through websites that list PCLs and ad-hoc searches. Applying inclusion and exclusion criteria resulted in the selection of 666 sources, which made relevant contributions related to PCLs in software engineering.

Results:

Conclusions:

ACM Computing Classification System (CCS)
Social and professional topics → Computing / technology policy → Intellectual property → Licensing

Avainsanat — Nyckelord — Keywords

open source, free / libre software, copyright, proprietary software, copyleft, license

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Software study track

Acknowledgements

much love to suvi, artemis, sami nurmivaara, prof männistö and prof mäntylä

dedicated to suvi <3

Contents

1 Introduction

PCLs play a central to the distribution of works in software engineering. For example in open source there must be an appropriate PCL attached to the source code in order for open-source software to be freely available for possible modification and redistribution. Because open source is central to software engineering the licenses enabling open source must also be considered important in the same context.

Public copyright license is defined by Wikipedia with the following words (Wikipedians, 2024a):

> "A PCL is a copyright license where the licensees are not limited. Examples include free content, open content, Creative Commons, free software and open source licences."

Understanding PCLs can be difficult. This could stem from the legal nature of the license texts and the large number of already-existing PCLs. The license texts usually favors correctedness over the readability for the developer. This is because the license text has to act as a valid legal instrument otherwise it cannot be endorsed (Ferguson, 2006). The lack of understanding of PCLs leaves too much room for interpretation. In June 21, 2023 International Business Machines' (IBM) Red Hat seemingly violated a PCL, the GNU General Public License version 2 (GPL-2.0) (Kuhn, 2023) (McGrath, 2023). This was an unpleasant surprise to the public since the project behind GNU General Public License (GPL), GNU Project initially attempted to ensure the users via the GPL have to the following three freedoms (GNU, 1996):

- Freedom 1: The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.
- Freedom2: The freedom to redistribute copies so you can help others
- Freedom 3: The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

Regardless, IBM's Red Hat essentially rendered the previously public Red Hat Enterprise

Linux (RHEL) into proprietary software. If the licenses would be more easily understood the proprietarization of RHEL would have been less of a surprise to the users.

On top of PCL details, software engineers in general have a tough time understanding the basic goals of PCLs used in software engineering. In the instance of the RHEL incident it would not have been a big surprise to software engineers if they would have known about other licenses and what they try to achieve or how old is GPLv2 and why it has been succeeded by GNU General Public License version 3 (GPL-3.0).

This thesis' goal is to contribute into the solving these problems in a structured manner. First we state definitions and terminology used in the scope of this thesis. We go over the reasons why there does not exist consistent terminology in this area and why the conversely the definitions are the most stabile ones in this area. Second we take a deep dive into the PCLs through a multivocal literature review. To make more information available, a mapping study connected to the terminology scope defined in the first step is needed. Third includes our own suggestions and basic knowledge for professionals and academics in the industry to enhance the understanding of PCLs in software engineering. This step also includes discussion of the future research and contributes to stablizing the terminology and reinforcing the already-existing definitions in the academic field.

1.1 Research goal, questions and contributions

The primary goal of this research is to conduct a multivocal literature review of the current state of PCLs in software engineering, the evaluation of the them and the evidence level of the research. The research aims to provide a novel perspective on relevant licenses and to extract key findings through a rigorous literature review process. The research questions of the review are:

- RQ1: How many PCLs in software engineering does there exist?
- RQ3: What is the average length of a PCL in software engineering?
- RQ3: What are the most common components seen in PCLs in software engineering?
- RQ4: What are the most common changes made to PCLS in software engineering?

Terms such as open source, source code, software freedom and other vocabulary must be defined in the scope of this thesis. Section 1.3 will examine this plethora of of terminology and definitions and will be used to establish a sound basis for discussing this broad subject.

This study has two main viewpoints. The first one is to provide rigorous multivocal research on PCLs to the academic field. Because this thesis already does the multivocal work on PCLs in software engineering the researches of the future can cite the results of this thesis without having to mark their study a multivocal one. This is the grand goal of this thesis. The second one is to provide insights and general metrics to the professional field of software engineering on PCLs. Hopefully this makes conversation on PCLs in software engineering easier and more rooted to scientific research rather than gut feeling and old, non-scientific articles on the insights and metrics of PCLs in software engineering.

1.2 Thesis structure

This thesis follows the IMRaD structure. Chapter 1 introduces the problem, this thesis' possible contributions and some further background. Chapter 2 goes over the process and the methods of the multivocal literature review. This is where most of the actual research takes place in. Chapter 3 presents results to the research questions. Chapter 4 discusses implications for research. The chapter also discusses software engineering professionals in the thesis' context and the validity of the thesis' research. Chapter 5 concludes this thesis with the help of the research questions and the future of the research.

1.3 Background and terminology of PCLs

The current terminology is used with different definitions which leads to inconsistencies in the field of software engineering. For example The Open Source Initiative (OSI) classifies GPL-3.0 under the term "open source" whereas the Free Software Foundation (FSF) classifies GPL-3.0 under the term "free software" (OSI, 2008)(Stallman, 2009). This is because their definitions on open source and free software differ from each other. Some parts of the two definitions are even mutually exclusive. This is rarely mentioned when people talk about Free and Open Source Software (FOSS) or Free / Libre and Open Source Software (FLOSS) which leads to misunderstanding that the two approaches are the same. This is why our focus will be PCLs in software engineering, which distinguishes our investigation from the broader topic of PCLs or the copyright law. This includs also PCLs that are not approved by the FSF nor OSI hence not falling under the group of FLOSS licenses. The term "copyleft" is defined by Mustonen, 2003 in the following way:

> "Copyeft is a novel licensing scheme. It facilitates open and decentralized software development. Its key feature is that once a program is licensed by the inventor, the subsequent programs based on the original must also be licensed similarly."

This is why the term is often used in the context of free software.

In this section we aim to increase the accessibility of our discussion by providing a concise overview of the background of the field of PCLs and the terms we employ.

To explain our emphasis on PCLs in software engineering, it is essential to examine the other possible areas of interest in PCLs. Our study classifies such efforts into eight domains as mentioned by the GNU Project (GNU, 2023).

These domains include:

- PCLs in software engineering
- PCLs in documentation for example architecture documentation of a project that may or may not be software or even publicly licensed
- PCLs in artistic works for example digital art, music or videos
- PCLs in educational works
- PCLs in fonts
- PCLs in viewpoints
- PCLs in physical objects
- PCLs in other works

The primary aim of this study is to investigate PCLs in software engineering process. However, it is important to acknowledge that PCLs in software engineering are only aspect of PCLs. These additional dimensions are crucial in adoption and implementation of PCLs in software engineering, but they are not the focus of this thesis.

For example, including artistic works such as music would require us to understand the basics of music theory and what sets apart distinct pieces of music from one another, something that could be outside the skillset of the author. While developing a comprehensive theory, framework, and tooling for PCLs as a whole is a gargantuan task beyond

the scope of a single thesis, narrowing our focus to software engineering enables us to examine a more concise and complete aspect of the main topic of this thesis.

As significant point of clarification, it is essential to acknowledge that PCLs are generally meant to be used as valid legal instruments. The question whether or not a PCL can act as a legal instrument is critical to the main function of these licenses. However, this thesis will not focus on the legal doctrine aspects either. The enforceability of PCLs has seen discussion in the academic field of law since the dawn of PCLs and since there's already an academic base for research it is likely the discussion seems to continue on with a healthy amount of activity (Duisburg, 2011).

Since the most recognized PCLs in software engineering in public are either open-source licenses or free-software licenses and since both paradigms are driven by different organizations with very different goals and values, it is understandable how non-standardized the terminology in the scope of PCL in SE is. The example given in the first section of this sub-chapter illustrates the challenges involved in maintaining consistency in the use of terminology in this emerging field and further warrants a closer inspection of the terminology to emphasize our own standing in the field.

To provide an understanding of the terminology used in this thesis, a Venn diagram is presented in Figure 1.1, which contextualizes the non-standardized terminology within the PCL scope as a whole. This perspective provides an increased understanding of where different subdomains fall in the larger picture of PCLs. Furthermore it is essential to note that PCLs in software engineering encompasses different aspects that require a closer examination.

Let us explore further the differences and similarities between open source and free software at the software engineering level of PCLs. This is a crucial step since we can see from the approximation in Figure 1.1 that the majority of PLCs are either free software, open source or both. We glanced over the free software definition in the first section of Chapter 1. Open Source Initiative defines open-source licenses in the Open Source Definition briefly in the following way (OSI, 2024):

> "Open source licenses are licenses that comply with the Open Source Definition - in brief, they allow software to be freely used, modified, and shared."

Like the FSF with free software, OSI has the final word on what passes as open source and what does not. For example a new software PCL will not classify as free software nor open source until the corresponding organization has acknowledged the software PCL as

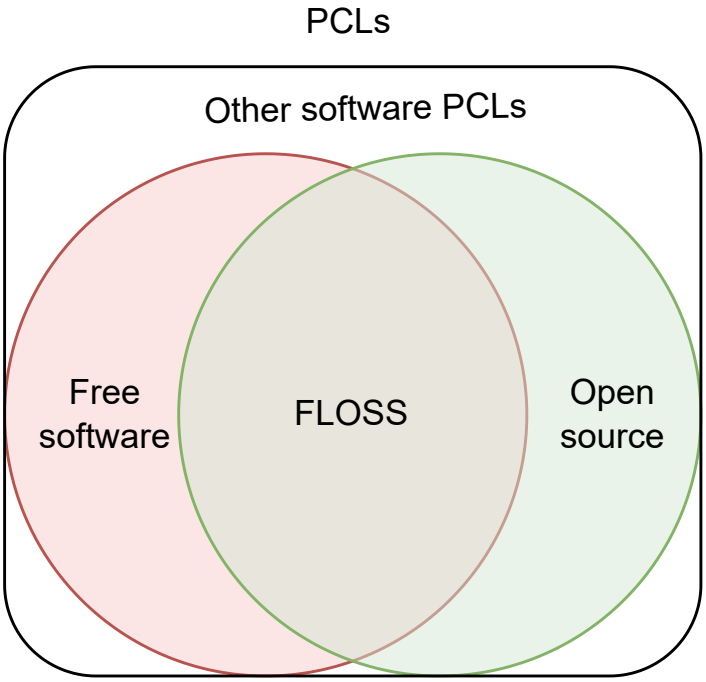

Figure 1.1: PCLs in software engineering

either free software, open source or neither. If a PCL is accepted by both FSF and OSI it will fall under the term FLOSS. If a PCL gets accepted by neither of the organization or it gets rejected by both organizations it will fall under other software PCLs in the Figure 1.1. In general free software license requirements are considered more strict than the open source license requirements. For the sake of perspective we could simplify the differences like so: free software requires the redistributions of the licensed software to be open as well but open source does not require this. The terms free software and open source are in general often misunderstood or just thought of as FLOSS collectively because the terms have a hard time conveying their paradigms in the natural language. One would not think free software does not mean software free of charge nor would one think that open source allows closed source redistributions of the licensed software. We will glance over the impacts on the industry of these two terms in Chapter 4.

With the context laid out in this chapter let us define PCLs in software engineering for the purpose of this study: Public software licenses are copyright licenses where the licensees are not limited and the copyright license in question is meant be used in licensing software source code. This helps us create the search strings and find the relevant literature for this thesis. This also helps us exclude PCLs regarding documentation, media and all other non-software targeted PCLs.

The quest to categorize every software PCL under some paradigm objectively is a complex one and cannot be comprehensively answered in a single paragraph. Therefore it is essential to continue taking the correct steps towards incresing the scientific understanding and providing the industry with examples, standards and processes to follow. However, as the following chapters reveal, a significant amount of effort is still being spent on solving the same problem multiple times, rather than building on existing knowledge and finding the next problem to solve. This thesis aims to contribute to mitigating this challenge by providing a rigorous analysis of the current state of the field. As the knowledge, conventions, and terminology take shape,we can look forward to reaching a state where less effort is spent on defining concepts and more on practical problem-solving.

2 Methods

This chapter aims to establish a precisely defined and rigorous research approach to enhance transparency and repeatability. We will take the steps required to ensure that every phase and decision is thoroughly documented, enabling the reader to retrace the research process. In a thesis made by a single researcher the lack of cross-examination of results with multiple researchers and the validation of evaluation criteria for opinion bias pose threats to validity, as will be clarified further in Chapter 4. Therefore, special attention will be paid to address these concerns. By following this approach, this research endeavors to contribute to the existing body of knowledge in the field of computer science in a robust and reliable manner.

The systematic literature review method (SLR) is a well-established approach for conducting a comprehensive and rigorous analysis of the existing research on specific research question or subject (Kitchenham and Charters, 2007). This paper presents a multivocal literature review (MLR). MLR is a SLR that includes both academic (AL) and grey literature (GL). This method was selected for this study to facilitate a thorough and scientifically interdisciplinary examination of PCLs in software engineering. The existing literature consists of PCLs and as such are considered gray literature, making the thesis a multivocal literature review.

This study follows the guidelines outlined by Kitchenham and Charters, 2007, to ensure its quality. The multivocal review method consists of three distinct phases: planning, conducting and reporting the review. This study stricly adhered to this structure. The phases can be further broken down into a research protocol, as illustrated in Figure 2.1. Adhering to the protocol is the first step in ensuring a well-documented and rigorous process, which increases the validity and auditability of the study.

The multivocal literature review process began with the formulation of research questions and the establishment of a comprehensive search strategy and scope. The search process was conducted by employing a quasi-gold standard (QGS) approach based on the implementation by Zhang and Ali Babar, 2010. After the completion of the search process, the inclusion and exclusion criteria were defined. To ensure a structured evaluation of the literature, a data extraction form was created. Finally, a strategy for analyzing the extracted data from the literature was designed.

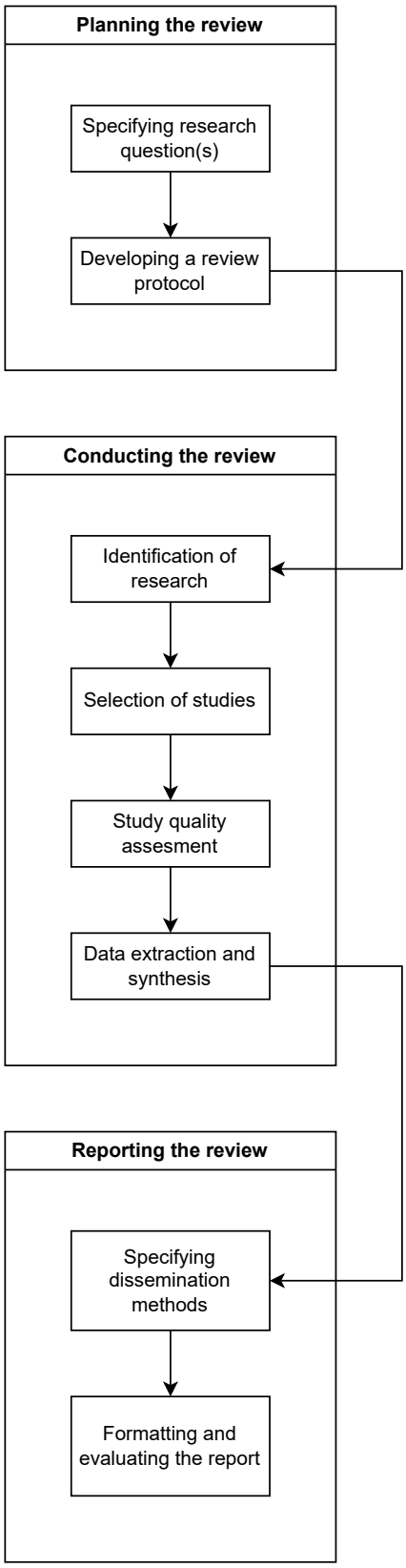

Figure 2.1: Three phases of a systematic literature review

To ensure the reliability and validity of the research protocol, it was validated against similar systematic literature reviews in computer science, the aforementioned guidelines by Kitchenham and Charters, 2007, and was further refined through an iterative process. Specifically, a subset of the data was tested on (The QGS) and any identified issues or problems were recorded and addressed. The details of this process are explained and thoroughly documented in the following sections. Similarly, the same approach was followed for the data extraction process, whereby a subset of literature was tested to refine the data extraction form. The revision of the form was undertaken as necessary to guarantee the completeness and accuracy of the extracted data.

2.1 Research questions

The research questions in this study served two primary purposes. Firstly, they aimed to provide an anaylsis of the existing multivocal literature on PCLs in software engineering for the researchers interested about the field. Secondly, the questions were designed to cater a secondary audience of professional software engineering practicioners. As discussed in the Chapter 1, the following research questions were addressed in this thesis:

- RQ1: How many PCLs in software engineering does there exist?
- RQ2: What is the average length of a PCL in software engineering?
- RQ3: What are the most common components seen in PCLs in software engineering?
- RQ4: What are the most common changes made to PCLS in software engineering?

The multivocal literature review in this thesis begins with addressing RQ1, which aims to provide the amount of PCLs that exist in software engineering. The review takes into account attributes like versions, supersedences to a different license family, formal or otherwise and recognizability. These attributes give us different amounts to existing PCLs in software engineering. This information could be most valuable for the practicioners out of all the research questions in the thesis since it could give some sense of the scale when picking a PCL that would serve the practicioners' needs the best.

Next RQ2 seeks to find the average length of the text of a PCL in software engineering. This research question has attributes like the number of characters, sentences, distinct sections and the size of the license on a computer screen. This information could be

valuable for the practicioners mentioned in the previous parapgrah for the same reasons of getting a better overview of the PCLs in software engineering. The research questions could also be beneficial for the practicioners working directly within the meta plane of PCLs in software engineering. Let us refer to the latter as researchers.

Finally RQ3 and RQ4 attempt to distinguish the top level paragraphs and other components of the PCLS in software engineering and what are the common reasons for the changes made to them throughout the years. The research questions go over the content of the changes and the implied and expressed reasons for making the changes. The answers to these last two research questions could again be useful for the researchers. The results can be used to introduce some notable background of the current PCLs in software engineering and enabling focus to more specific areas inside this PCLs in software engineering.

2.2 Search stragey

The search process was conducted on various PCL listing websites. The selection criteria for the literature were defined after the search process and the selection process was based on inclusion and exclusion criteria. The inclusion and exclusion criteria and each step of exclusion on the literature found was documented and is available as Appendix A. The used criteria are presented later in this chapter.

The data extraction process was performed in a standardized and systematic manner, with the aim of obtaining all relevant information from the selected literature. The data extraction form used included information such as license name, release year, text length and inferred purpose and is available Table 2.2. The extracted data was then used to answer the research questions and perform the data analysis. The results of the data analysis were then reported in a rigorous manner.

2.2.1 Search method

The search was conducted on various PCL listing websites, as mentioned earlier, to obtain a broad set of multivocal literature. This approach yielded a large number of literature that were processed to a subset of high-relevance literature using exclusion and quality criteria presented later in this chapter. Manual searching of databases with thousands of PCLs is not feasible, and it is prone to researcher bias and may overlook relevant venues from other scientific disciplines. However, a preliminary manual search was performed

Field	Value
Publisher	Massachusetts Insitute of Technology
SPDX identifier	MIT
Debian FSG compatible	Yes
FSF approved	Yes
OSI approved	Yes
GPL compatible	Yes
Copyleft	No
Linking from code with a different license	Yes

Table 2.1: MIT License Wikipedia page infobox

to reduce the number of iterations required and establish the quasi-gold standard (QGS) mentioned earlier.

2.2.2 Search scope and terms

Originally the search terms would have been present just like in a normal MLR or SLR. Keywords however produced highly varying and non-reproducabe results in Google Scholar and Google Search. Some PCL listing websites such as FSF's list of pages categorized as licenses could not be found from Google Search even with the `site` operator: `site:https://directory.fsf.org/wiki/Category:License`. Although the page has been up since 2013 for some reason Google has not crawled the page in 10 years (FSF, 2024). Hence why this thesis does not include search terms per se.

Instead, for establishing a QGS we started defining our search scope from the Wikipedia page of one of the most used open source license according to Balter, 2015, the MIT license (Wikipedians, 2024b). The infobox contained fields in the order shown in Table 2.1.

As we defined PCLs in software engineering as copyright licenses where the licensees are not limited and the copyright license in question is meant be used in licensing software source code in Chapter 2 and our research questions focus on finding measurements and reasonings to the PCLs' various attributes, we decided to gather PCLs from the related web pages of the aforementioned categorizers: SPDX, FSF, OSI and GNU. The publisher, GPL compatibility, copyleft and the linking exception did not result in any meaningful PCL listing websites. This leaves us with the SPDX, Debian FSG compatibility, FSF and

OSI from which all resulted in some sort of PCL listing websites.

With the search for the initial PCL listing websites completed we moved onto the search process itself.

2.3 Search process

The literature selection process was divided into multiple stages, as outlined in Figure 2.2. The initial step involved the formation of the first PCL listing websites through which the first literature would be acquired from.

In the first stage, the search was conducted using the ”SPDX License List” (Linux Foundation, 2024), ”The DFSG and Software Licenses” (Debian, 2024), FSF’s ”Category:License” Wiki page (FSF, 2024), GNU’s ”Various Licenses and Comments about Them” (GNU, 2023) and "OSI Approved licenses" (OSI, 2024). The initial list of PCLs excluding duplicates is provided in Appendix A

In the second stage, the inclusion and exclusion criteria were applied to further filter the literature and reduce the number of licenses to be reviewed. This involved a manual review of the full licenses. The exclusion reason as a shortcode (e.g. I1 = failed to meet inclusion criteria 1 or E2 = met exclusion criteria 2) is provided in Appendix B

The third stage was the most time-consuming and involved a manual review of the full licenses. After reading and evaluating each license, a final round of exclusions was completed and documented. The remaining licenses were used for data collection and analysis in the final part of the study. The final list of licenses is available in Appendix C.

2.4 Inclusion and exclusion criteria

To be eligible for the data collection and analysis, a license had to meet all of the following inclusion criteria:

- I1: The license focuses mostly on the copyright of software source code
- I2: inclusion criteria 2

Additionally, licenses were excluded if they met any of the following criteria:

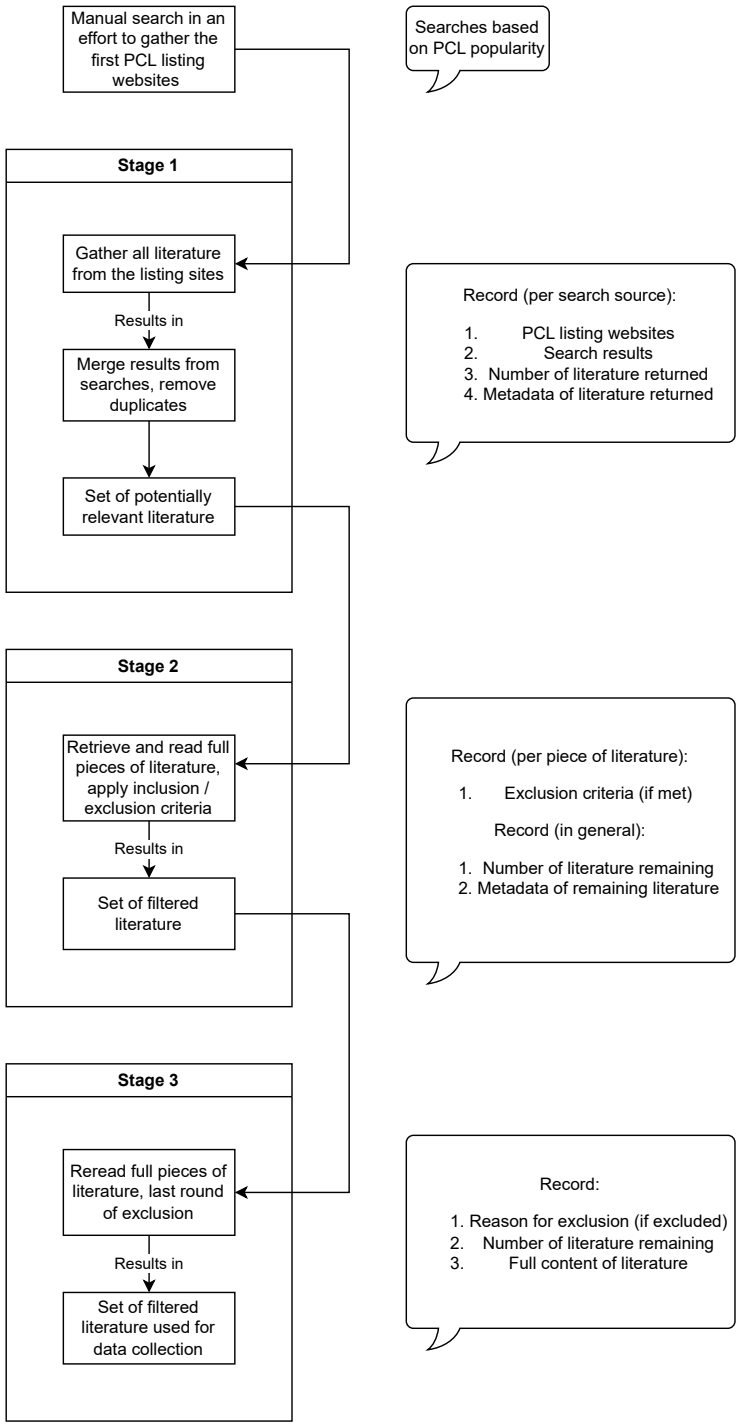

Figure 2.2: Search process divided into stages

#	Field	Concern/Research question
F1	Name	Documentation
F2	Length	RQ3
F3	FSF approval	Documentation
F4	OSI approval	Documentation
F5	Inferred purpose	RQ1, RQ2, RQ4

Table 2.2: Data extraction form

- E1: The piece of literature is a license exception
- E2: exclusion criteria here
- E3: exclusion criteria here
- E4: exclusion criteria here

The relevance of each piece of literature was evaluated based on inclusion and exclusion criteria stated above. In cases where there was doubt about the suitability of a license, a more in-depth manual examination of its content was performed. The reason for exclusion was documented for each license that failed to meet the criteria, and when it was unclear, the license was included by default.

Another relevant criteria related to the ones of inclusion and exclusion are the quality and evidence criteria. These criteria used by Dybå et al., 2007 were not put into practice in this thesis since individual PCLs per se might not be meaningful in a results, evidence nor quality perspective. This puts more emphasis on the inclusion and exclusion criteria so that is something we must be mindful about.

2.5 Data collection and data analysis

To answer the research questions of this thesis, a thorough examination of the selected primary literature was conducted and the necessary data was collected using data extraction form presented in Table 2.2. A record of extracted data was kept for anaysis and is available as Appendix B.

The subsequent chapter presents the outcomes of the steps taken in the study, as discussed above.

week 21 thu: why should i mention threats to validity in two places? ah never mind i guess the section about validity in ch2 was just about saying it out loud that validity must be evaluated and watched upon. hmm the qgs thingy seems very shallow but ig everything can be fixed later on as long as i document everything i find during the process of writing the thesis.

week 21 fri: i think i'll be fine as long as i just keep writing stuff. i can correct and re-check titles and topic alignments later on, f.ex. the qgs and kitchenham2007. one threat to validity is the way i have constructed the google sheets. i used xlookup on each consequtive sheet to check if the currently checked spdx identifier is also existing in one of the sheets before and if so just fill the full name of the license. example: =xlookup(B2; 'SPDX licenses'!B:B; 'SPDX licenses'!A:A; "not found"). why this is a threat to validity is because one could imagine a scenario where DFSGLicenses would have introduced a license with the SPDX identifier "MIT" yeah sure it matches the MIT license on SPDX sheet from before but because we haven't read the full license from DFSG we can't know for sure if they meant for example MIT License (MIT), MIT No Attribution (MIT-0), Enlightenment License (e16) [MIT-advertising], CMU License (MIT-CMU), enna License (MIT-enna), feh License (MIT-feh), MIT Festival Variant (MIT-Festival), MIT License Modern Variant (MIT-Modern-Variant), MIT Open Group variant (MIT-open-group), MIT testregex Variant (MIT-testregex), MIT Tom Wu Variant (MIT-Wu), MIT +no-false-attribs license (MITNFA) or in the worst case something completely different like MIT-DFSG-edition. this is however something we have to live with since as of now there are way too many licenses in the first search stage for the scope of this thesis for the author to check them all individually. what should you do with fsf:license:other?

week 22 mon: we are not going to include fsf:license:other. for example babl seems to be in that category but is actually licensed under gplv3.

week 22 tue: definitely not including fsf:license:other. there are a whopping 5282 programs whose license fsf just hasn't put up yet. i don't know what im going to do when i have to review the full licenses. for stage 3 licenses i should create a spreadsheet that doesn't allow duplicate attribute-having licenses. ok do stage inside spreadsheet as well and so that there is a simple clickable url to the license text. wayback machine in the end all of the urls via api or something.

3 Results

This chapter employes the data extracted from the set of primary literature, available as Appendix A, utilizing the methods outlined in Chapter 2 to address the research questions. Firstly, a summary of the general statistics collected and aggregated from the studies is presented. Following that, an analysis of the data is performed to provide answers to each of the research questions.

how many licenses and why

statistical overview with figures (mapping study)

how many licenses during each stage (figure)

basic statistic on final licenses (figure)

essential statistics (figure)

3.1 Placeholder question (RQ1)

figures and literature identifier tables

3.2 Placeholder question (RQ2)

figures and literature identifier tables

3.3 Placeholder question (RQ3)

figures and literature identifier tables

3.4 Placeholder question (RQ4)

figures and literature identifier tables

4 Discussion

indications

follow-up observation

observation 1

observation 2

sum-up from those two

4.1 Implications for research

how to improve scientific scene 1

how to improve scientific scene 2

how to improve scientific scene 3

4.2 Implications for software engineering professionals

how to improve professional scene 1

how to improve professional scene 2

how to improve professional scene 3

overall

4.3 Limitations and threats to validity

major limitation

possible threats to validity

4.3.1 Limitations of license selection for review

efforts to inclusion

as with all slr all licenses cannot be reviewed manually

license selection was done in sufficient manner

4.3.2 Limitations in data extraction

importance of data extraction

lack of measurements and tooling

5 Conclusions

primary objective of this study

conclusions from each rq

5.1 Future research

adopting a clear baseline

Docker CLA, SSPL

make cla easier maybe with gpg / joplin easy cla sign

LICENSE highlighting.js

what kind of efforts and why

what this thesis has provided

Bibliography

Balter, B. (2015). *Open source license usage on GitHub.com - The GitHub Blog*. https://web.archive.org/web/20240409120258/https://github.blog/2015-03-09-open-source-license-usage-on-github-com/. Accessed: 2024 April 9.

Debian (2024). *The DFSG and Software Licenses*. https://web.archive.org/web/20240415104532/https://wiki.debian.org/DFSGLicenses. Accessed: 2024 April 15.

Duisburg, H. von (2011). "The Enforceability of the General Public License in the US". In: *Zeitschrift der Deutsch-Amerikanischen Juristen-Vereinigung e.V. 36 (2011)* 2, pp. 69–70. URL: https://heinonline.org/HOL/Page?handle=hein.journals/dajvnws2011&id=75.

Dybå, T., Dingsøyr, T., and Hanssen, G. (Oct. 2007). "Applying Systematic Reviews to Diverse Study Types: An Experience Report". In: pp. 225–234. ISBN: 978-0-7695-2886-1. DOI: 10.1109/ESEM.2007.59.

Ferguson, D. (2006). "Syntar Errors: Why Version 3 of the GNU General Public License Needs Debugging". In: *North Carolina Journal of Law & Technology* 7.2, pp. 397–420. URL: https://heinonline.org/HOL/Page?handle=hein.journals/ncjl7&id=403.

FSF (2024). *Category:License - Free Software Directory*. https://web.archive.org/web/20240409111815/https://directory.fsf.org/wiki/Category:License. Accessed: 2024 April 9.

GNU (1996). *What is Free Software?* https://web.archive.org/web/19980126185518/https://www.gnu.org/philosophy/free-sw.html. Accessed: 2024 Feb 8.

– (2023). *Various Licenses and Comments about Them*. https://web.archive.org/web/20240226115940/https://www.gnu.org/licenses/license-list.html. Accessed: 2024 Feb 26.

Kitchenham, B. and Charters, S. (Jan. 2007). "Guidelines for performing Systematic Literature Reviews in Software Engineering". In: 2.

Kuhn, B. M. (2023). *A Comprehensive Analysis of the GPL Issues With the Red Hat Enterprise Linux RHEL Business Model*. https://web.archive.org/web/20240205080551/https://sfconservancy.org/blog/2023/jun/23/rhel-gpl-analysis/. Accessed: 2024 Feb 5.

Linux Foundation (2024). *SPDX License List*. https://web.archive.org/web/20240412103557/https://spdx.org/licenses/. Accessed: 2024 April 12.

McGrath, M. (2023). *Furthering the evolution of CentOS Stream*. https://web.archive.org/save/https://www.redhat.com/en/blog/furthering-evolution-centos-stream. Accessed: 2024 Feb 5.

Mustonen, M. (2003). "Copyleft—the economics of Linux and other open source software". In: *Information Economics and Policy* 15.1, pp. 99–121. ISSN: 0167-6245. DOI: https://doi.org/10.1016/S0167-6245(02)00090-2. URL: https://www.sciencedirect.com/science/article/pii/S0167624502000902.

OSI (2008). *GNU General Public License version 3*. https://web.archive.org/web/20240226075409/https://opensource.org/license/gpl-3-0. Accessed: 2024 Feb 26.

– (2024). *Licenses - Open Source Initiative*. https://web.archive.org/web/20240307121412/https://opensource.org/licenses. Accessed: 2024 March 7.

Stallman, R. (2009). "Viewpoint Why "open source" misses the point of free software". In: *Commun. ACM* 52.6, pp. 31–33. ISSN: 0001-0782. DOI: 10.1145/1516046.1516058. URL: https://doi.org/10.1145/1516046.1516058.

Wikipedians (2024a). *Category:PCLs — Wikipedia, The Free Encyclopedia*. https://web.archive.org/web/20240131085240/https://en.wikipedia.org/wiki/Category:Public_copyright_licenses. Accessed: 2024 Jan 31.

– (2024b). *MIT License - Wikipedia*. https://web.archive.org/web/20240411081352/https://en.wikipedia.org/wiki/MIT_License. Accessed: 2024 April 11.

Zhang, H. and Ali Babar, M. (2010). "On searching relevant studies in software engineering". In: *Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering*. EASE'10. UK: BCS Learning & Development Ltd., pp. 111–120.

Appendix A Primary literature identified in the search process, duplicates removed

Literature identifier	Name	Year
L1	MIT License	1987
L2	Apache 2.0	666

Table A.1: A list of literature and the basic filtering step.

Appendix B Primary literature reviewed, read in full and inclusion/exclusion criteria applied

Literature identifier	Name	Year
L1	MIT License	1987
L2	Apache 2.0	666

Table B.1: List of literature with the inclusion/exclusion criteria applied