Master's thesis

Master's Programme in Computer Science

Public copyright licenses in Software Engineering: A Multivocal Literature Review

Akira Taguchi

March 21, 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	March 21, 2024	18 pages, 3 appendix pages

Tiivistelmä — Referat — Abstract

Public copyright licenses (PCL) play a major part 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. sub-problem here. thesis' contribution to solution here.

Tell about the research method here.

Tell about the results here.

Tell about the discussion here.

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 and prof männistö

dedicated to suvi <3

Contents

1 Introduction

PCLs play a major part 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 (Category:PCLs, 2012):

> "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 multivocal literature through a systematic 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 secondary 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 are there in software engineering?
- RQ2: What is the average length of a PCL in software engineering?
- RQ3: How often do PCLs in software engineering change?
- RQ4: How have PCLs in software engineering changed?
- RQ5: Why do PCLs in software engineering get new versions?

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 goals: 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.

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. 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).

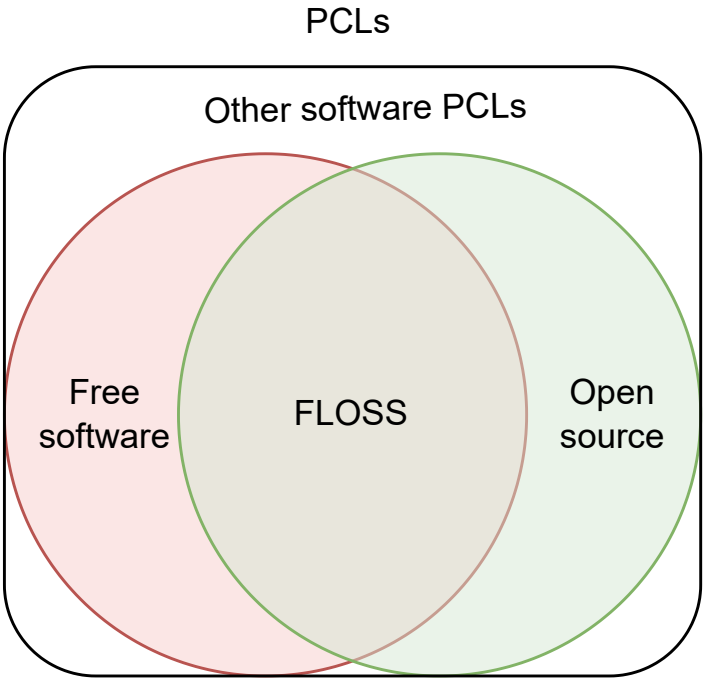

Figure 1.1: PCLs in software engineering

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 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 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 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. The method of a systematic literature review is still conducted the same way.

This study follows the guideslines outlined by Kitchenham and Charters, 2007, to ensure its quality. The systematic 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 systematic 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, and a strategy was developed for assessing the quality of the multivocal literature that met these criteria. 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.

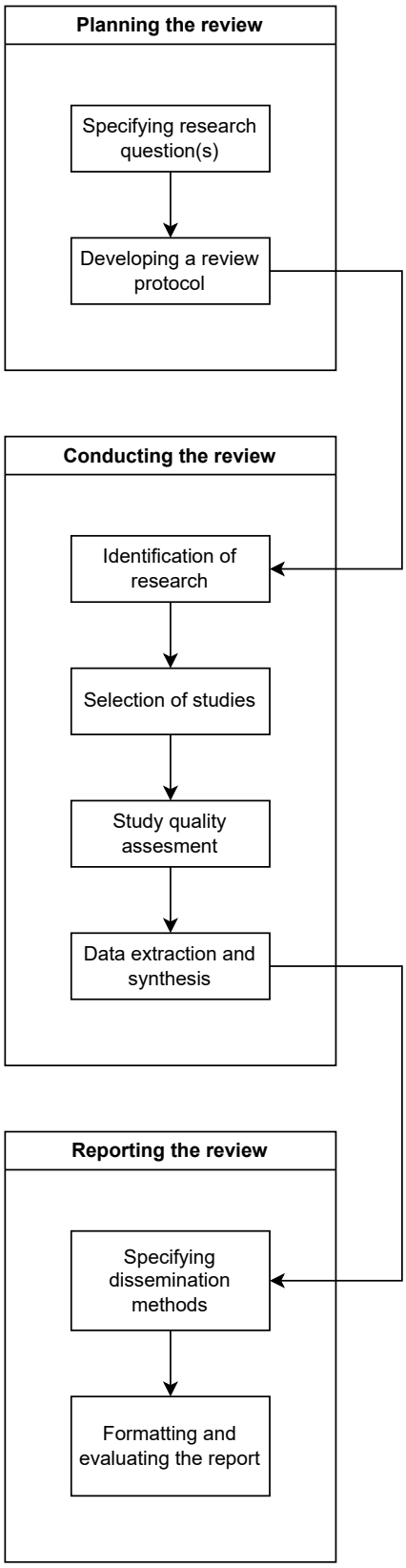

Figure 2.1: Three phases of a systematic literature review

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 are there in software engineering?
- RQ2: What is the average length of a PCL in software engineering?
- RQ3: How often do PCLs in software engineering change?
- RQ4: How have PCLs in software engineering changed?
- RQ5: Why do PCLs in software engineering get new versions?

The systematic 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 RQs 3 to 5 review changes related to PCLs in software engineering. The research questions go over the average length of time between new changes to the licenses and their corresponding versions, the content of the changes and the implied and expressed reasons for making the changes. The answers to these three 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 in Google Search. 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 2.1. 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 Google Search, 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 to reduce the number of iterations required and establish the quasi-gold standard (QGS) mentioned earlier.

2.2.2 Search scope and terms

The search terms for this study were determined through an iterative process that took into account the research questions and topic. Synonyms for key terms were included and combined using Boolean logic to form a comprehensive search string. The search function on the search engine Google Search served as the foundation for developing the search string. Further specification on Google Search is available online (Google, 2024).

The search string was established on a basis of a quasi-gold standard as proposed by Zhang and Ali Babar, 2010. For establishing a QGS we employed a manually crafted search string based on the topic and research questions of this study. 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 manually formulated the search string:

2.3 Search process

2.4 Inclusion and exclusion criteria

2.5 Quality and evidence criteria

2.6 Data collection and data analysis

#	Field	Concern/Research question
F1	Name	Documentation
F2	Year	RQ3, RQ4, RQ5
F3	Length	RQ2
F4	Inferred purpose	RQ1, RQ5
F3	FSF approval	Documentation
F4	OSI approval	Documentation

Table 2.1: Data extraction form

3 Results

3.1 Placeholder question (RQ1)

3.2 Placeholder question (RQ2)

3.3 Placeholder question (RQ3)

4 Discussion

4.1 Implications for research

4.2 Implications for software engineering professionals

4.3 Limitations and threats to validity

4.3.1 Limitations of license selection for review

4.3.2 Limitations in data extraction

5 Conclusions

5.1 Future research

Bibliography

Category:PCLs (2012). *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.

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.

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.

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.

Google (2024). *Google Search - What Is Google Search And How Does It Work*. https://www.google.com/search/howsearchworks/. Accessed: 2024 March 20.

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.

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.

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.

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 and their inclusion/exclusion criteria

Appendix B Primary literature reviewed in the quality criteria step and manual exclusions reasons

Appendix C Primary literature reviewed, read in full and data extracted