

Version 0.88.5

A Practical Guide for Developers, Managers, OS Experts, and Companies

Open Source License Compendium

How to Achieve Open Source License Compliance*

Karsten Reincke[†] Greg Sharpe[‡]

6th March 2013

*) This text is licensed under the Creative Commons Attribution-ShareAlike 3.0 Germany License (<http://creativecommons.org/licenses/by-sa/3.0/de/>): Feel free “to share (to copy, distribute and transmit)” or “to remix (to adapt)” it, if you “[...] distribute the resulting work under the same or similar license to this one” and if you respect how “you must attribute the work in the manner specified by the author(s) [...]”):

In an internet based reuse please mention the initial authors in a suitable manner, name their sponsor *Deutsche Telekom AG* and link it to <http://www.telekom.com>. In a paper-like reuse please insert a short hint to <http://www.telekom.com>, to the initial authors, and to their sponsor *Deutsche Telekom AG* into your preface. For normal citations please use the scientific standard.

[derived from myCsrF (= 'mind your Scholar Research Framework') ©K. Reincke CC BY 3.0 <http://mycsrffodina.de/>]

[†]) Deutsche Telekom AG, Products & Innovation, T-Online-Allee 1, 64295 Darmstadt

[‡]) Deutsche Telekom AG, Telekom Deutschland GmbH, Landgrabenweg, Bonn

The Open Source Community is a swarm: it is more powerful than a set of arbitrarily selected experts. We are proud to have its support. Gladly we thank (in alphabetical order):

Eitan Adler,
Stefan Altmeyer,
John Dobson,
Steffen Härtlein,
Ta'Id Holmes,
Michael Kern,
Michael Machado,
Thorsten Müller,
Oliver Podebradt,
Thomas Quiehl,
Peter Schichl,
Helene Tamer,
Bernhard Tsai,
and all the others...

Contents

1	Introduction	8
2	Open Source: The Same Idea, Different Licenses	13
2.1	The protecting power of the Affero Gnu Public License (AGPL) [tbd]	23
2.2	The protecting power of the Apache License (ApL)	23
2.3	The protecting power of the BSD licenses	25
2.4	The protecting power of the Eclipse Public License (EPL)	26
2.5	The protecting power of the European Union Public License (EUPL)	28
2.6	The protecting power of the Gnu Public License (GPL) [tbd]	30
2.7	The protecting power of the Lesser Gnu Public License (LGPL) [tbd]	30
2.8	The protecting power of the MIT license	30
2.9	The protecting power of the Mozilla Public License (MPL)	31
2.10	The protecting power of the Microsoft Public License (MS-PL)	33
2.11	The protecting power of the Postgres License (PgL)	34
2.12	The protecting power of the PHP License	35
2.13	Summary	35
3	Open Source: About Some Side Effects	38
3.1	The problem of implicitly releasing patents	38
3.1.1	ApL statements concerning patents	42
3.1.2	EPL statements concerning patents	43
3.1.3	EUPL statements concerning patents	44
3.1.4	GPL statements concerning patents	44
3.1.4.1	GPL-2.0	44
3.1.4.2	GPL-3.0	45
3.1.5	LGPL statements concerning patents	46
3.1.5.1	LGPL-2.1	46
3.1.5.2	LGPL-3.0	47
3.1.6	MPL statements concerning patents	47
3.1.7	MS-PL statements concerning patents	48
3.2	Excursion: What is a 'Derivative Work' - the basic idea of open source	48
3.3	Excursion: The problem of license compatibility [tbd]	50
3.4	Excursion: open source software and money [tbd]	50
4	Open Source Use Cases: Concept and Taxonomy	51
4.1	Overview of the OSUC classes and tokens	54
4.2	The OSUC taxonomy	55
4.3	About a suppressed degree of complexity	56
5	Open Source Use Cases: Find the License Fulfilling To-do Lists	57
5.1	A standard form for gathering the relevant information	58
5.2	The taxonomic Open Source Use Case Finder	59
5.3	The open source use cases and its to-do list references	61

Contents

6	Open Source License Compliance: To-Do Lists	67
6.1	Some general remarks on 'giving' someone a file	67
6.2	Apache licensed software	68
6.2.1	ApL-1: Using the software only for yourself	68
6.2.2	ApL-2: Passing the unmodified software as source code	69
6.2.3	ApL-3: Passing the unmodified software as binaries	70
6.2.4	ApL-4: Passing a modified program as source code	70
6.2.5	ApL-5: Passing a modified program as binary	71
6.2.6	ApL-6: Passing a modified library as independent source code	72
6.2.7	ApL-7: Passing a modified library as independent binary	73
6.2.8	ApL-8: Passing a modified library as embedded source code	74
6.2.9	ApL-9: Passing a modified library as embedded binary	75
6.2.10	Discussions and Explanations [tbd]	76
6.3	BSD licensed software	76
6.3.1	BSD-1: Using the software only for yourself	77
6.3.2	BSD-2: Passing the unmodified software as source code	78
6.3.3	BSD-3: Passing the unmodified software as binary	78
6.3.4	BSD-4: Passing a modified program as source code	79
6.3.5	BSD-5: Passing a modified program as binary	80
6.3.6	BSD-6: Passing a modified library as independent source code	80
6.3.7	BSD-7: Passing a modified library as independent binary	81
6.3.8	BSD-8: Passing a modified library as embedded source code	82
6.3.9	BSD-9: Passing a modified library as embedded binary	82
6.3.10	Discussions and Explanations	84
6.4	MIT licensed software	86
6.4.1	MIT-1: Using the software only for yourself	86
6.4.2	MIT-2: Passing the unmodified software	86
6.4.3	MIT-3: Passing a modified program	87
6.4.4	MIT-4: Passing a modified library independently	88
6.4.5	MIT-5: Passing a modified library as embedded component	88
6.4.6	Discussions and Explanations	89
6.5	Microsoft Public License	90
6.5.1	MS-PL-1: Using the software only for yourself	90
6.5.2	MS-PL-2: Passing the unmodified software	91
6.5.3	MS-PL-3: Passing a modified program as source code	91
6.5.4	MS-PL-3: Passing a modified program as binary	92
6.5.5	MS-PL-4: Passing a modified library independently as source code	93
6.5.6	MS-PL-4: Passing a modified library independently as binary	93
6.5.7	MS-PL-5: Passing a modified library as embedded source code	94
6.5.8	MS-PL-5: Passing a modified library as embedded binary	95
6.5.9	Discussions and Explanations	95
6.6	Postgres Licensed Software in the usage context of ... [tbd]	96
6.6.1	PGL specific use case 1	96
6.6.2	PGL specific use case n	97
6.7	PHP Licensed Software in the usage context of ... [tbd]	97
6.7.1	PHP specific use case 1	97
6.7.2	PHP specific use case n	97
6.8	Eclipse Licensed Software in the usage context of ... [tbd]	98
6.8.1	EPL specific use case 1	98
6.8.2	EPL specific use case n	98

Contents

6.9	European Union Public Licensed Software in the usage context of ... [tbd]	99
6.9.1	EUPL specific use case 1	99
6.9.2	EUPL specific use case n	99
6.10	Mozilla Public Licensed Software in the usage context of ... [tbd]	99
6.10.1	MPL specific use case 1	99
6.10.2	MPL specific use case n	100
6.11	LGPL Licensed Software in the usage context of ... [tbd]	100
6.11.1	LGPL specific use case 1	100
6.11.2	LGPL specific use case n	100
6.12	AGPL Licensed Software in the usage context of ... [tbd]	100
6.12.1	AGPL specific use case 1	100
6.12.2	AGPL specific use case n	101
6.13	GPL Licensed Software in the usage context of ... [tbd]	101
6.13.1	GPL specific use case 1	101
6.13.2	GPL specific use case n	102
7	Open Source Licenses and Their Legal Environments [tbd]	103
8	Conclusion	104
9	Appendices	105
9.1	Some Additional Remarks on the OSLiC Quotation Style	105
9.2	Some Widespread Open Source Myths	106
9.2.1	Why	108
9.2.2	What	117
	Periodicals, Shortcuts, and Abbreviations	119
	Bibliography	121

Contents

Table 0.1: History of the Open Source License Compendium

2013-03-07	0.90.1	CeBIT release ▷ to-do lists for the most important licenses added ▷ branches merged and new master published
2013-02-16	0.8.90	CeBIT pre release ▷ new arguing structure focused on the topic license fulfillment ▷ new classifying license review ▷ new top down introduction
2012-12-28	0.8.0	internal EOY release ▷ many distributed improvements unified in branch kreinck
2012-08-25	0.5.2	expanded break through release ▷ MIT license fulfilling to-do lists ▷ using integrated Eclipse spell checking methods
2012-07-06	0.4.0	break through release ▷ open source use case definition and taxonomy ▷ open source use case based general finder ▷ corresponding BSD specific mini finder ▷ BSD license fulfilling to-do lists
2012-03-22	0.2.1	▷ framework published as first community edition
2012-01-31	0.1.8	▷ renamed existing introduction as prolegomena ▷ inserted a shorter top-down written introduction ▷ inserted an OSLiC disclaimer
2012-01-21	0.1.7	▷ oscCopiedButNotRead.bib expanded ▷ list of periodicals and shortcuts added
2011-12-29	0.1.6	▷ many bibliographic data added
2011-10-17	0.1.5	▷ bibliographic data updated
2011-09-29	0.1.4	▷ document history integrated ▷ typos erased
2011-09-28	0.1.3	▷ review of english teacher integrated
2011-09-19	0.1.2	▷ first comments of english teacher integrated
2011-09-15	0.1.1	▷ improvements of John integrated
2011-09-12	0.1.0	▷ introduction completed: purpose and methods

Disclaimer

This book shall be thoroughly developed – together with the open source community. Finally, it shall deliver reliable information based on the insight that the swarm knows more than the single fish.

But nevertheless, the OSLiC can not offer more than the opinion(s) of its authors and contributors. It is only one voice of the chorus discussing the open source licenses. For protecting the authors and contributors from charges and claims of indemnification we adopt the lightly modified GPL3 disclaimer:

THERE IS NO WARRANTY FOR THE OSLiC, TO THE EXTENT PERMITTED BY APPLICABLE LAW. THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE TEXT “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE OSLiC IS WITH YOU. SHOULD THE OSLiC PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE OSLiC AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE OSLiC (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE OSLiC TO COOPERATE WITH ANY OTHER TOOLS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Particularly, it must be highlighted that - referred to your solitary case - the OSLiC can not and shall not replace a legal review or a legal advice by lawyers. It may inspire developers, managers, open source experts, and companies to find good solutions which they finally should let be reviewed by legal counselors¹.

¹⁾ For German readers: The OSLiC naturally respects the German ‘Rechtsdienstleistungsgesetz’. It only contains legal reflections addressed to a general public. The OSLiC may only be read as an “nur an die Allgemeinheit gerichtete Darstellung und Erörterung von Rechtsfragen”.

1 Introduction

This chapter briefly describes the idea behind the OSLiC, the way it should be used and the way it can be read – which is indeed not quite the same.

This book focuses on just one issue: *What needs to be done in order to act in accordance with the licenses of those open source software we use?* The *Open Source License Compendium* aims at reliably answering this question – in a simple and easy to understand manner. However, it is not just another book on *open source* in general². The intention is, rather, for it to be a tool for simplifying the activities for achieving license conformity.

This compendium was created out of a necessity at *Deutsche Telekom AG* and a challenge for some of its software developers and project managers: Of course, the company itself wants to behave as license compliantly as its employees. Unfortunately, they could not find a reference text which simply lists what precisely must be done in order to comply with the license of that piece of open source being used.

As some of these co-workers in Telekom projects, even we – the initial authors of the OSLiC – did not want to become open source license experts only for being able to use open source software in accordance with the respective licenses. We did not want to become lawyers. We just wanted to do more efficiently, what in those days claimed much time and many resources. We were searching for clear guidance instead of having to determine a correct way through the jungle of open source licenses – over and over again, project for project. We loved using the high-quality open source software to improve our performance. We liked using

²⁾ Meanwhile, there are tons of literature dealing with open source. Trying to expand your knowledge by means of books and articles might let you get lost in literature: our list of secondary literature may adumbrate this 'danger of being overwhelmed'. But nevertheless, our bibliography at the end of the OSLiC is not complete. Moreover, it is not intended to be complete. It is only an extract representing the background information we did not directly cite in the OSLiC. If we were forced to indicate two books for attaining a good overview on the topic of *open source (licenses)* we would name (a) the 'Rebel Code' (for a German version cf. *Moody, Glyn*: Die Software-Rebellen. Die Erfolgsstory von Linus Torvalds und Linux; transl. from the American [edition, 2000] by Annemarie Pumpering; Landsberg am Lech: verlag moderne industrie, 2001, ISBN 3-478-38730-2, passim – for an English version cf. *Moody, Glyn*: Rebel Code: Linux And The Open Source Revolution; [New York]: Basic Books, 2002, ISBN 978-0738206707, p. passim) and (b) the 'legal basic conditions' (cf. *Jaeger, Till a. Axel Metzger*: Open Source Software. Rechtliche Rahmenbedingungen der Freien Software; 3rd edition. München: Verlag C.H. Beck, 2011, passim). But fortunately, we are not forced to do so.

1 Introduction

it legally. But we did not like to laboriously discuss the legal constraints of the many and different open source licenses.

What we needed, was an easy-to-use handout which would lead us without any detours to executable lists of work items. We wished to obtain to-do lists, tailored to our usecases and our licenses. We needed reliable lists of tasks we only had to execute for being sure that we were acting in accordance with the open source license. When we started out, such a compendium did not exist.

For solving this problem our company took three decisions:

The first decision our company arrived at, was to support a small group of employees to act as *a board of open source license experts*: They should offer a service for the whole company. Projects, managers, and developers should be able to ask this board what they have to do for complying with a specific open Source License under specific circumstances. And this board should answer with authoritative to-do lists whose executions would assure that the requestors are acting according to the corresponding open source licenses. The idea behind this decision was simple. It would save cost and increase quality if one had one central group of experts instead of being obliged to select (and to train) developers – over and over again, for every new project. So, the *OSRB* – the *Telekom Open Source Review Board* – was founded as an internal expert group – as a self-organizing, bottom-up driven community.

The second decision our company took, was to allow this *Telekom OSRB* to collect their results systematically – in the form of a reusable compendium. The idea behind this decision was also simple: The more the internal service became known, the more the workload would increase: the more work, the more resources, the more costs. So, such a compendium should save costs and enable the requestors to find answers by themselves without becoming license experts: For all default cases, they should find an answer in the compendium instead of having to request that their work is analyzed by the OSRB. Thus, the planned *Telekom Open Source License Compendium* prevents Telekom from being forced to increase the number of OSRB members in the future.

The third decision our company reached, was to allow the *Telekom OSRB* to create the compendium in the same mode of cooperation that open source projects usually use. Again, a simple reason evoked this ruling: If in the future – as a rule – not a reviewing OSRB, but a simple manual should assure the open source license compliant behavior of projects, programmers and managers, this book had of course to be particularly reliable. There is a known feature of the open source working model: the ongoing review by the cooperating community increases the quality. Therefore, the decision, not only to write an internal 'Telekom hand-out', but to enable the whole community to use, to modify and to redistribute a broader *Open Source License Compendium*, was a decision for improving quality. Consequently, the *OSRB* decided to publish the *OSLiC* as a set of LaTeX

1 Introduction

sources, publicly available via the open repository github³. And it licensed the OSLiC under Creative Commons Attribution-ShareAlike 3.0 Germany License⁴.

But to publish the *OSLiC* as a free book has another important connotation – at least for the *Telekom OSRB*: It is also intended to be an appreciative *giving back* to the *open source community* which has enriched and simplified the life of so many employees and companies over so many years.

Altogether, the OSLiC follows five principles:

To-do lists as the core, discussions around them : Based on a simple form to gather information concerning the concrete use of a piece of open source software and its license, the OSLiC shall offer an easy-to-use finder taking the requestor to the respective to-do list for ensuring license conformity. In addition, all these elements of the OSLiC should comprehensibly be introduced and discussed without disturbing the usage itself.

Quotations with thoroughly specified sources : The OSLiC shall be revisable and reliable. It shall comprehensibly argue and explicitly specify why it adopts which information, from whom, in which version, and why⁵.

No clearing the forest, but cutting a swath : The OSLiC has to deal with li-

³) Get the code by using the link <https://github.com/dtag-dbu/oslic>; get project information by <http://dtag-dbu.github.com/oslic/> or by <http://www.oslic.org/>.

⁴) This text is licensed under the Creative Commons Attribution-ShareAlike 3.0 Germany License (<http://creativecommons.org/licenses/by-sa/3.0/de/>): Feel free “to share (to copy, distribute and transmit)” or “to remix (to adapt)” it, if you “[...] distribute the resulting work under the same or similar license to this one” and if you respect how “you must attribute the work in the manner specified by the author(s) [...]”: In an internet based reuse please mention the initial authors in a suitable manner, name their sponsor *Deutsche Telekom AG* and link it to <http://www.telekom.com>. In a paper-like reuse please insert a short hint to <http://www.telekom.com>, to the initial authors, and to their sponsor *Deutsche Telekom AG* into your preface. For normal citations please use the scientific standard.

⁵) For that purpose, we are using an ‘old-fashioned’ bibliographic style with footnotes, instead of endnotes or inline-hints. We want to enable the users to review or to ignore our comments and hints just as they prefer – but on all accounts without being disturbed by large inline comments or frequent page turnings. We know that modern writer guides prefer less ‘noisy’ styles (pars pro toto cf. *MLA*: *MLA Handbook for Writers of Research Papers*; 7th edition. New York: The Modern Language Association of America, 2009, ISBN 978–1–60329–024–1, passim). But for a reliable usage – challenged by the often modified internet sources – these methods are still a little imprecise (for details → OSLiC, pp. 105. For a short motivation of the style used in the OSLiC cf. *Reincke, Karsten*: *Classical Scholar Texts With Footnotes based on LaTeX, BibTeX, Koma, jurabib and mykeds-CSR*; 2012 (URL: <http://www.fodina.de/en/closedprojects/latex-addons/classical-scholar.html>) – reference download: 2013-02-10, passim. For a more elaborated legitimizing version cf. *Reincke, Karsten*: *(Geistes-) Wissenschaftliche Texte mit jurabib. Dienst am Leser, Dienst am Scholaren: Über Anmerkungsapparate in Fußnoten - aber richtig.* [n.l.], 2012 (URL: <http://download.fodina.de/fodinaClassicalScholarFoNoDe.pdf>) – reference download: 2013-02-10, passim).

1 Introduction

censes and their legal aspects, no doubt. But it shall not discuss all details of every aspect. It shall focus on one possible way to act according to a license in a specific usecase – even it is known that there might be alternatives⁶.

Take the license text seriously : The OSLiC shall not give general lectures on legal discussions, much less shall it participate in them. It shall only find one dependable way for each license and each usecase to comply with the license. The main source for this analysis shall be the exact reading of the open source licenses themselves – based on and supported by the interpretation of benevolent lawyers and rational arguing software developers. The OSLiC shall respect that open source licenses are written for software developers (and sometimes by developers).

Trust the swarm : The OSLiC shall be open for improvements and adjustments encouraged and stimulated also by other people than employees of *Deutsche Telekom AG*.

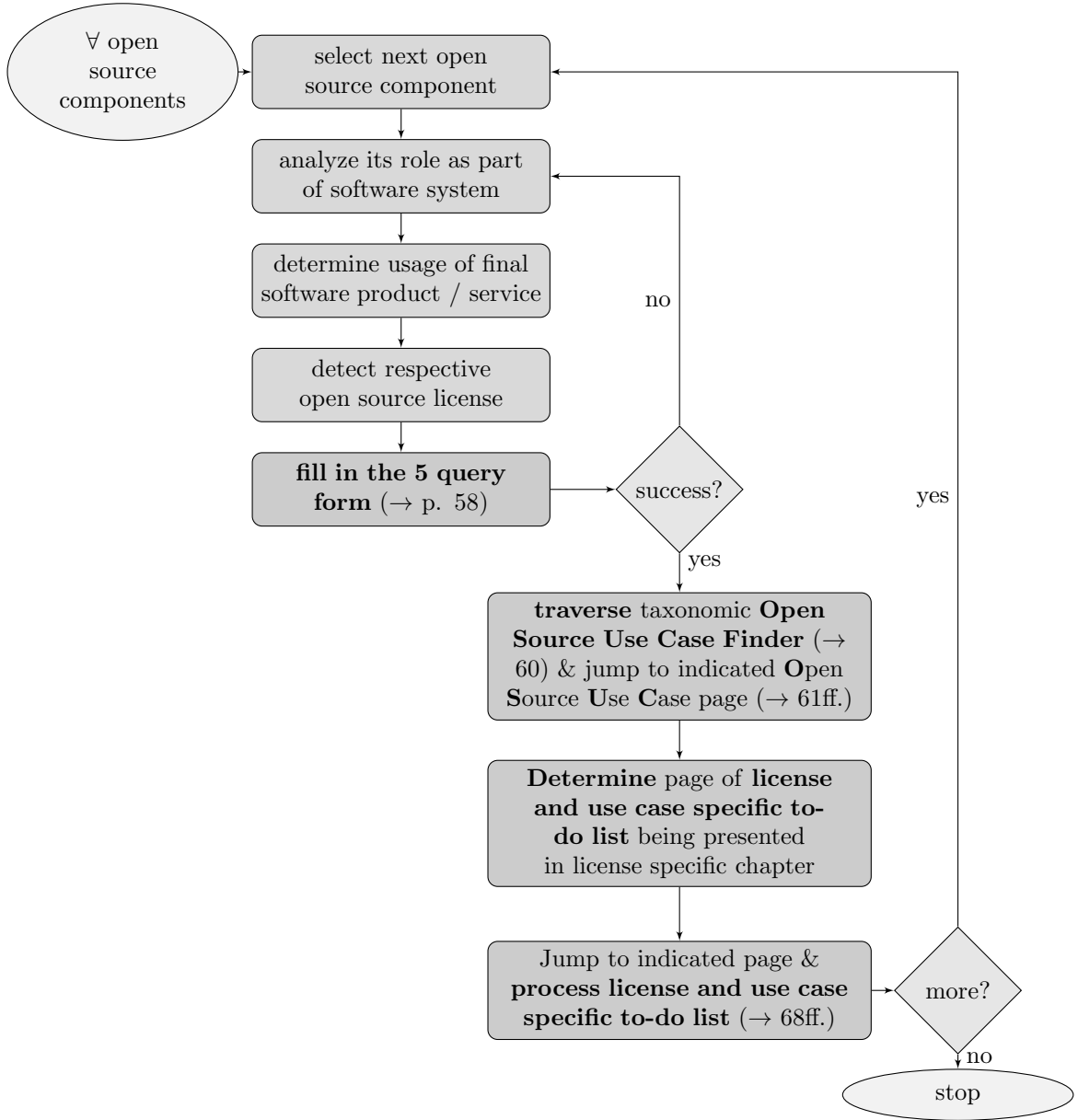
Based on these principles the OSLiC offers two methods for being used:

First and foremost the readers expect to simply and quickly find those to-do lists fitting their needs. Here is the respective process⁷:

⁶) The OSLiC shall not counsel projects with respect to their specific needs. This must remain the task for lawyers and legal experts. The OSLiC cannot and shall not replace a legal review or a legal advice by lawyers. It shall inspire developers, managers, open source experts, and companies to find good solutions which they finally should have reviewed by legal counselors in the end. For the German readers let us repeat again: The OSLiC naturally respects the German 'Rechtsdienstleistungsgesetz'. It only contains legal reflections addressed to a general public. Its content may only be read as a "nur an die Allgemeinheit gerichtete Darstellung und Erörterung von Rechtsfragen".

⁷) For the well known 'quick and dirty hackers' – as we tend to be, too – we have integrated a shortcut: If you already know the license of the open source package you want to use and if you are very familiar with the meaning of the open source use cases we defined, then you might directly jump to the corresponding license specific chapter, without 'struggling' with *OSLiC 5 query form* (→ OSLiC p. 58), the taxonomic *Open Source Use Case Finder* (→ 60) or the *Open Source Use Case* page (→ 61ff.): Some of the chapters dedicated to specific open source licenses starts with a license specific finder offering a set of license specific use cases – which according to the complexity of the license – in some cases could be stripped down. But the disadvantage of this method is that you have to apply your knowledge about the use cases and their side effects by yourself without the systematically guiding OSLiC process.

1 Introduction



Second, the readers might wish to comprehend the whole analysis. So, we briefly discuss open source licenses taxonomies as the basis for a license compliant behavior⁸. We consider some side effects of acting according to the open source licenses⁹. Finally, we study the structure of open source use cases¹⁰.

So, let us close our introduction by using, modifying, and (re)distributing a well known wish of a well known man: Happy (Legally) Hacking.

⁸) → OS LIC “Open Source: The Same Idea, Different Licenses”, pp. 13

⁹) → OS LiC “Open Source: About Some Side Effects”, pp. 38

¹⁰) → OS LiC “Open Source Use Cases: Concept and Taxonomy”, pp. 51

2 Open Source: The Same Idea, Different Licenses

This chapter describes different license models which follow the common idea of free open source software. We want to discuss existing ways of grouping licenses to underline the limits of building such clusters: These groups are often used as 'virtual prototypic licenses' which shall deliver a simplified view at the conditions how to act according to the respective real license instances. But one has to meet the requirements of a specific license, not one's own generalized idea of a set of licenses. Nonetheless, also we offer a new structuring view into the world of the open source licenses. We will use a novel set of grouping criteria by referring to the common intended purpose of licenses: each license is designed to protect something or someone against something or someone. Following this pattern, we can indeed summarize all Open Source Licenses in a comparable way.

Grouping open source licenses is commonly done. Even the set of the *open source licenses*¹¹ itself is already a cluster being established by a set of grouping criteria: The “distribution terms” of each software license that intends to become an open source license, “[...] must comply with the [...] criteria” of the *Open Source Definition*¹², maintained by the *Open Source Initiative*¹³ and often abbreviated as *OSD*. So, this *OSD* demarcates ‘the group of [potential] open source licenses’ against ‘the group of not open sources licenses’¹⁴.

Another way to cluster the *Free Software Licenses* is specified by the “Free Software Definition”. This *FSD* contains four conditions which must be met by any free software license: any *FSD* compliant license must grant “the freedom to run a program, for any purpose [...]”, “the freedom to study how it works, and adapt it to (one’s) needs [...]”, “the freedom to redistribute copies [...]”, and finally “the freedom to improve the program, and release your improvements [...]”¹⁵. Surprisingly this definition implies that the requirement *the sourcecode*

¹¹⁾ cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted; 2012 [n.y.] [⟨URL: http://opensource.org/licenses/alphabetical⟩](http://opensource.org/licenses/alphabetical) – reference download: 2013-01-22, wp.

¹²⁾ cf. *Open Source Initiative*: The Open Source Definition; 2012 [n.y.] [⟨URL: http://www.opensource.org/docs/osd⟩](http://www.opensource.org/docs/osd) – reference download: 2012-06-21, wp.

¹³⁾ cf. *Open Source Initiative*: The Open Source Initiative; 2012 [n.y.] [⟨URL: http://www.opensource.org/about/⟩](http://www.opensource.org/about/) – reference download: 2013-01-22, wp.

¹⁴⁾ More precisely: meeting the *OSD* is only a necessary condition for becoming an *open source license*. The sufficient condition for becoming an *open source license*, is the approval by the *OSI* which offers a process for the official approval of *open source license* (cf. *Open Source Initiative*: The [OSI] Licence Review Process; 2012 [n.y.] [⟨URL: http://www.opensource.org/approval⟩](http://www.opensource.org/approval) – reference download: 2013-01-22, wp.).

¹⁵⁾ cf. *Stallman, Richard M.*: Free Software Definition; originally written in 1996; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, p. 41.

2 Open Source: The Same Idea, Different Licenses

must be openly accessible, is 'only' a derived condition. If the "freedom to make changes and the freedom to publish improved versions" shall be "meaningful", then the "access to the source code of the program" is a prerequisite. "Therefore, accessibility of source code is a necessary condition for free software."¹⁶

The difference between the OSD and the FSD has often been described as a difference of emphasis¹⁷: Although both definitions "[...] (cover) almost exactly the same range of software", the *Free Software Foundation* – as it is said – "prefers [...] (to emphasise) the idea of freedom [...]" while the *OSI* wants to underline the philosophically indifferent "development methodology"¹⁸.

A third method to group of free software and free software licenses is specified by the "Debian Free Software Guideline" which is embedded into the "Debian Social Contract". This "DFSG" contains nine defining criteria which – as Debian itself says – have been "[...] adopted by the free[sic!] software community as the basis of the Open Source Definition"¹⁹.

A rough understanding of these methods might result in the conclusion that these three definitions are extensionally equal and only differ intensionally. But that is not true. To unveil the differences, let us compare the clusters *OSI approved licenses*, *OSD compliant licenses*, *DFSG compliant licenses*, and *FSD compliant licenses* extensionally, by asking whether they *could* establish different sets of licenses²⁰.

¹⁶⁾ cf. *Stallman*: Free Software Definition, 1996, p. 41.

¹⁷⁾ This is also the viewpoint of Richard M. Stallman: On the one hand, he clearly states that the "Free Software movement" and the "open source movement" generally "[...] disagree on the basic principles, but agree more or less on the practical recommendations" and that he "[...] (does) not think of the open source movement as an enemy". On the other hand, he delineates the two movements by stating that "for the open source movement, the issue of whether software should be open source is a practical question, not an ethical one", while "for the Free Software movement, non-free software is a social problem and free software is the solution" (cf. *Stallman, Richard M.*: Why 'Free Software' is Better than 'Open Software'; originally written in 1998; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, p. 55). Consequently, Richard M. Stallman summarizes the positions in a simple way: "[...] 'open source' was designed not to raise [...] the point that users deserve freedom". But he and his friends want "to spread the idea of freedom" and therefore "[...] stick to the term 'free software'" (id., l.c., p. 59). For a brush-up of this position, expressing again that "(o)pen source is a development methodology [and that] free software is a social movement" with an "ethical imperative" cf. *Stallman, Richard*: Viewpoint: Why "Open Source" Misses the Point of Free Software; in: Communications of the ACM, 52 June (2009), No. 6 (URL: <http://doi.acm.org/10.1145/1516046.1516058>) – reference download: 2011-12-29, p. 31

¹⁸⁾ pars pro toto: cf. *Fogel, Karl*: Producing Open Source Software; How to Run a Successful Free Software Project; Beijing, Cambridge, Köln [...]: O'Reilly, 2006, ISBN 978-0-596-00759-1, p. 232.

¹⁹⁾ cf. *Debian*: The Debian Free Software Guidelines (DFSG); 2013 [n.y.] (URL: http://www.debian.org/social_contract#guidelines) – reference download: 2013-01-22, p. wp.

²⁰⁾ Indeed, for analyzing the extensional power of the definition we have to regard all potentially

2 Open Source: The Same Idea, Different Licenses

First, the difference most easy to determine is that of an unidirectional inclusion: By definition, the *OSI approved licenses* and the *OSD compliant licenses* meet the requirements of the OSD²¹. But only the *OSI approved licenses* have successfully passed the OSI process²² and therefore are officially listed as *open source licenses*²³. Hence, on the one hand, *OSI approved licenses* are *open source licenses* and vice versa. On the other hand, both – the *OSI approved licenses* and the *open source licenses* – are *OSD compliant licenses*, but not vice versa.

Second, a similar argumentation allows to distinguish the *DFSG compliant licenses* from the *OSI approved licenses*. As it is stated, the OSD “[...] is based on the Debian Free Software Guideline and any license that meets one definition almost meets the other”²⁴. But then again, meeting the definition is not enough for being an official open source license: the license has to be approved by the OSI²⁵. Thus, it follows that all *OSI approved licenses* are also *DFSG compliant licenses*, but not vice versa.

Third, – by ignoring the “few exceptions” which have appeared “over the years”²⁶ – it can be said that, because of their ‘kinsmanlike’ relation, at least the *OSD compliant licenses* are also *DFSG compliant licenses* and vice versa.

Last but not least, it must be stated that the (potential) set of free software licenses must be greater than all the other three sets: On the one side, the FSD requires that a license of free software must not only allow to read the software, but must also permit to use, to modify, and to distribute it²⁷. These conditions are covered by at least the first three paragraphs of the OSD concerning the topics “Free Redistribution”, “Source Code”, and “Derived Works”²⁸. On the other side, the OSD contains at least some requirements which are not mentioned by the FSD and which nevertheless must be met by a license in order to be qualified as an OSD compliant license²⁹. It follows then that there may exist licenses which fulfill all conditions of the FSD and nevertheless do not fulfill at least some conditions of the OSD³⁰. So, the set of all (potential) *Free Software Licenses* must

covered licenses, not only the already existing licenses, because the subset of really existing licenses still could be expanded by developing new licenses which fit the definition.

²¹⁾ cf. *Open Source Initiative*: The Open Source Definition, 2012, wp.

²²⁾ cf. id., ibid.

²³⁾ cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.

²⁴⁾ cf. *Fogel*: Producing Open Source Software, 2006, p. 233.

²⁵⁾ cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.

²⁶⁾ cf. *Fogel*: Producing Open Source Software, 2006, p. 233.

²⁷⁾ cf. *Stallman*: Free Software Definition, 1996, p. 41.

²⁸⁾ cf. *Open Source Initiative*: The Open Source Definition, 2012, wp.

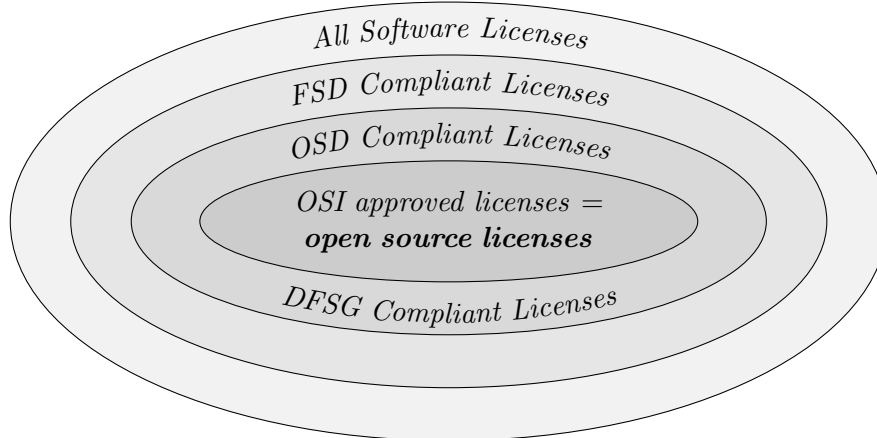
²⁹⁾ For example, see the condition that “the license must be technology-neutral” (cf. id., ibid.).

³⁰⁾ Again: we must consider the extensional potential of the definitions, not the set of really existing licenses. In this context, it is irrelevant that actually all existing Free Software Licenses like GPL, LGPL or AGPL indeed are also classified as open source licenses. We are referring to the fact that there might be generated licenses which fulfill the FSD, but not

2 Open Source: The Same Idea, Different Licenses

be greater than the set of all (potential) *open source licenses* and greater than the set of *OSD compliant licenses*.

All in all, we can visualize the situation as follows:



It should be clear without longer explanations that these clusters don't allow to extrapolate to the correct compliant behaviour according to the *open source licenses*: On the one hand, all larger clusters do not talk about the *open source licenses*. On the other hand, the *open source license cluster* itself only collects its elements on the base of the OSD which does not stipulates concrete license fulfilling actions for the licensee.

The next level of clustering *open source licenses* concerns the inner structure of these *OSI approved licenses*. Even the OSI itself has recently discussed whether a better kind of grouping the listed licenses would better fit the needs of the visitors of the OSI site³¹. And finally the OSI ends up in the categories “popular and widely used (licenses) or with strong communities”, “special purpose licenses”, “other/miscellaneous licenses”, “licenses that are redundant with more popular licenses”, “non-reusable licenses”, “superseded licenses”, “licenses that have been voluntarily retired”, and “uncategorized licenses”³².

Another way to structure the field of open source licenses is to think in “types of open source licenses” by grouping the “*academic licenses*, named as such because they were originally created by academic institutions”³³, the “*reciprocal licenses*”,

the OSD.

³¹⁾ cf. *Open Source Initiative*: OSI Mailing List. License-discuss. Draft of new OSI licenses landing page; 2012 [n.y.] (URL: <http://projects.opensource.org/pipermail/license-discuss/2012-April/000332.html>) – reference download: 2013-01-29, wp.

³²⁾ cf. *Open Source Initiative*: Open Source Licenses by Category; 2013 [n.y.] (URL: <http://opensource.org/licenses/category>) – reference download: 2013-01-29, wp.

³³⁾ cf. *Rosen, Lawrence*: Open Source Licensing. Software Freedom and Intellectual Property Law; Upper Saddle River, New Jersey: Prentice Hall PTR, 2005, ISBN 0-13-148787-6, p.69.

2 Open Source: The Same Idea, Different Licenses

named as such because they “[...] require the distributors of derivative works to distribute those works under same license including the requirement that the source code of those derivative works be published”³⁴, the “*standard licenses*”, named as such because they refer to the reusability of “industry standards”³⁵, and the “*content licenses*”, named as such because they refer to “[...] other than software, such as music art, film, literary works” and so on³⁶.

Both kinds of taxonomy directly help to find the relevant licenses which should be used for new (software) projects. But again: none of these categories allow to infer the license compliant behaviour, because the categories are mostly defined based on license external criteria: whether a license is published by a specific kind of organization or whether a license deals with industry standards or other kind of works than software, inherently do not evoke a license fulfilling behaviour.

Only the act of grouping into the “*academic licenses*” and the “*reciprocal licenses*” touches the idea of license fulfilling doings, if one – as it has been done – expands the definition of the “*academic licenses*” by the specification that these licenses “[...] allow the software to be used for any purpose whatsoever with no obligation on the part of the licensee to distribute the source code of derivative works”³⁷. With respect to this additional specification, the clusters “*academic licenses*” and the “*reciprocal licenses*” indeed might be referred as the “main categories” of (open source) licenses³⁸: By definition, they are constituting not only a contrary, but contradictory opposite. However, it must be kept in mind that they constitute an inherent antagonism, an antinomy inside of the set of open source licenses³⁹.

Connatural to the clustering into *academic licenses* and *reciprocal licenses* is the grouping into *permissive licenses*, *weak copyleft licenses*, and *strong copyleft licenses*: Even Wikipedia already uses the term “permissive free software licence” in the meaning of “a class of free software licence[s] with minimal requirements about how the software can be redistributed” and “contrasts” them with the “copyleft licences” as those “with reciprocity / share-alike requirements”⁴⁰.

³⁴) cf. *Rosen*: Open Source Licensing, 2005, p. 70.

³⁵) cf. id., *ibid.*

³⁶) cf. id., l.c., p. 71.

³⁷) cf. id., *ibid.*

³⁸) cf. id., l.c., p. 179.

³⁹) Hence, it is at least a little confusing to say that “the open source license (OSL) is a reciprocal license” and “the Academic Free License (AFL) is the exact same license without the reciprocity provisions” (cf. id., l.c., p. 180): If the BSD license is an AFL and if an AFL is not an OSL and if the OSI approves only OSLs, then the BSD license can not be an approved open source license. But in fact, it still is (cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp).

⁴⁰) cf. *Wikipedia (en)*: Permissive free software licence; n.l., 2013 [n.y.] <URL: http://en.wikipedia.org/wiki/Permissive_free_software_licence> – reference download: 2013-02-02, wp.

2 Open Source: The Same Idea, Different Licenses

Some other authors name the set of *academic licenses* the “permissive licenses” and specify the *reciprocal licenses* as “restrictive licenses”, because in this case – as a consequence of the embedded “copyleft” effect – the source code must be published in case of modifications. They additionally introduce the subset of “strong restrictive licenses” which additionally require that an (overarching) derivative work must be published under the same license⁴¹. The next refinement of such clustering concepts directly uses the categories “[open source] licenses with a strict copyleft clause”⁴², “[open source] licenses with a restricted copyleft clause”⁴³, and “[open source] licenses without any copyleft clause”⁴⁴. Finally, this viewpoint can directly be mapped to the categories *strong copyleft* and *weak copyleft*: While on the one hand, “only changes to the weak-copylefted software itself become subject to the copyleft provisions of such a license, [and] not changes to the software that links to it”, on the other hand, the “strong copyleft” states “[...] that the copyleft provisions can be efficiently imposed on all kinds of derived works”⁴⁵.

Based on this approach to an adequate clustering and labeling⁴⁶, we can develop the following picture:

⁴¹⁾ pars pro toto cf. *Buchtala, Rouven*: Determinanten der Open Source Software-Lizenzwahl. Eine spieltheoretische Analyse; Frankfurt am Main, Berlin, Bern [... etc.]: Peter Lang, 2007 (= Informationsmanagement und strategische Unternehmensführung), [Vol./No.] 12), ISBN 978-3-631-57114-9, p. 57.

⁴²⁾ Originally stated as “Lizenzen mit einer strengen Copyleft-Klausel”. Cf. *Jaeger a. Metzger*: Open Source Software. Rechtliche Rahmenbedingungen der Freien Software, 2011, p. 24.

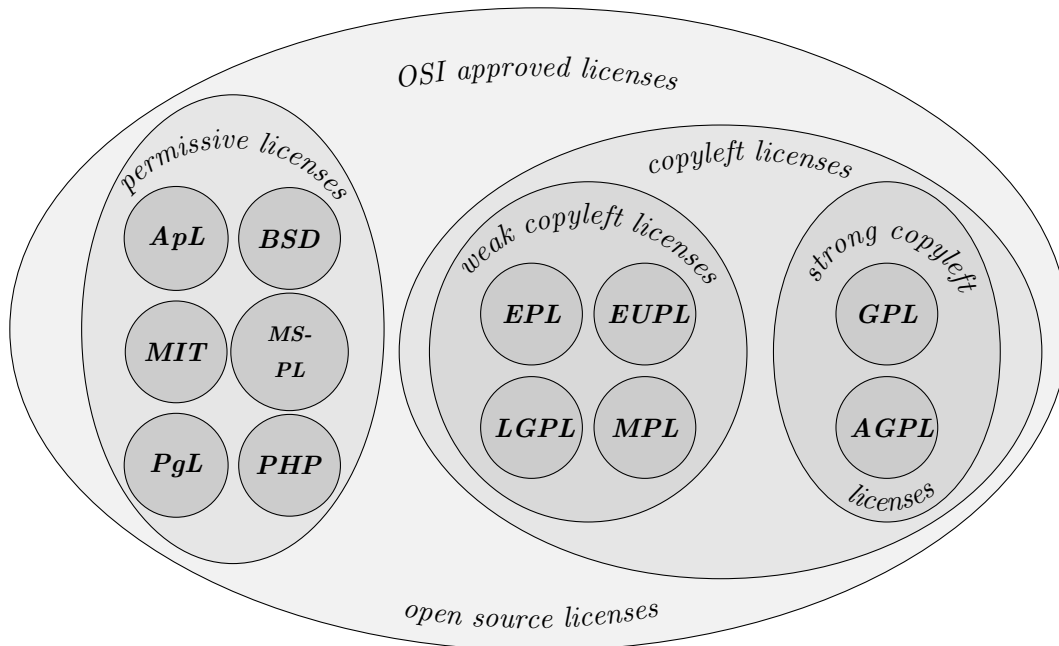
⁴³⁾ Originally stated as “Lizenzen mit einer beschränkten Copyleft-Klausel”. Cf. *id.*, l.c., p. 71.

⁴⁴⁾ Originally stated as “Lizenzen ohne Copyleft-Klausel”. Cf. *id.*, l.c., p. 83.

⁴⁵⁾ cf. *Wikipedia (en)*: Copyleft; n.l., 2013 [n.y.] <URL: <http://en.wikipedia.org/wiki/Copyleft>> – reference download: 2013-02-02, wp.

⁴⁶⁾ Finally, we should also mention that there still exists other classifications which might become important in other contexts. For example, the ifross license subsumes under the main category “Open Source Licenses” the subcategories “Licenses without Copyleft Effect”, “Licenses with Strong Copyleft”, “Licenses with Restricted Copyleft”, “Licenses with Restricted Choice”, or “Licenses with Privileges” – and let finally denote these categories also licenses which are not listed by the OSI (cf. *ifross*: License Center; 2011 [n.y.] <URL: http://www.ifross.org/ifross_html/lizenzcenter-en.html> – reference download: 2013-02-26, wp.). This is well reasonable if one refers to the meaning of the OSD (cf. *Open Source Initiative*: The Open Source Definition, 2012, wp). The OSLiC wants to simplify its object of study by referring to the approved open source licenses (cf. *Open Source Initiative*: The [OSI] Licence Review Process, 2012, wp) listed by the OSI (cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp).

2 Open Source: The Same Idea, Different Licenses



This extensionally based clarification of a possible open source license taxonomy is probably well-known and often – more or less explicitly – referred to⁴⁷. Unfortunately, this taxonomy still contains some misleading underlying messages:

Permissive has a very positive connotation. So, the antinomy of *permissive licenses* versus *copyleft licenses* implicitly signals, that the *permissive licenses* are in any meaning better, than the *copyleft licenses*. Naturally, this 'conclusion' is evoked by confusing the extensionally definition and the intensional power of the labels. But that is the way we – the human beings – like to think.

Anyway, this underlying message is not necessarily 'wrong'. It might be convenient for those people or companies who only want to use open source software without being restricted by the *obligation to give something back* as it has been introduced by the 'copyleft'⁴⁸. But there might be other people and companies who emphasize the protecting effect of the copyleft licenses. And indeed,

⁴⁷) Even the FSF itself uses the term 'permissive non-copyleft free software license' (pars pro toto: cf. *Free Software Foundation: Various Licenses and Comments about Them*; 2013 [n.y.] <URL: <http://www.gnu.org/licenses/license-list.html>> – reference download: 2013-02-08, wp/section 'Original BSD license') and contrasts it with the terms 'weak copyleft' and 'strong copyleft' (pars pro toto: cf. id., l.c., wp/section 'European Union Public License')

⁴⁸) De facto, *copyleft* is not *copyleft*. Apart from the definition, its effect depends on the particular licenses which determine the conditions for applying the copyleft 'method'. For example, in the GPL, the copyleft effect is bound to the criteria 'being distributed'. Later on, we will collect these conditions systematically (see chapter *Open Source Use Cases: Concept and Taxonomy*, pp. 51). Therefore, here we still permit ourselves to use a somewhat 'generalizing' mode of speaking.

2 Open Source: The Same Idea, Different Licenses

at least the open source license⁴⁹ *GPL*⁵⁰ has initially been developed to protect the freedom, to enable the developers to help their “neighbours” and to get the modifications back⁵¹: So, “Copyleft” is defined as a “[...] method for making a program free software and requiring all modified and extended versions of the program to be free software as well”⁵². It is a method⁵³ by which “[...] the code and the freedoms become legally inseparable”⁵⁴. Because of these disparate interests of hoping not to be restricted and hoping to be protected, it could be helpful to find a better label – an impartial name for the cluster of *permissive licenses*. But until that time, we should at least know that this taxonomy still contains an underlying declassing message.

The other misleading interpretation is – counter-intuitively – evoked by using the concept of ‘copyleft licenses’. By referring to a cluster of *copyleft licenses* as the opposite of the *permissive licenses*, one implicitly also sends two messages: First, that republishing one’s own modifications is sufficient to comply with the *copyleft licenses*. And, secondly, that the *permissive licenses* do not require anything to be done for obtaining the right to use the software. Even if one does not wish to evoke such an interpretation, we – the human beings – tend to take the things as simple as possible⁵⁵. But because of several aspects, this understanding of the

⁴⁹) Although RMS naturally prefers to specify it as a *Free Software License* (s. p. 14)

⁵⁰) As the original source cf. *Free Software Foundation*: GNU General Public License, version 2; 1991 [n.y. of the html page itself] (URL: <http://www.gnu.org/licenses/gpl-2.0.html>) – reference download: 2013-02-05, wp. Inside of the OSLiC, we constantly refer to the license versions which are published by the OSI, because we are dealing with officially approved open source licenses. For the ‘OSI-GPL’ cf. *Open Source Initiative*: GNU General Public License, version 2 (GPL-2.0). Version 2, June 1991; 1991 [n.y. of the html page itself] (URL: <http://opensource.org/licenses/GPL-2.0>) – reference download: 2013-02-05, wp

⁵¹) The history of the GNU project is multiply told. For the GNU project and its initiator cf. pars pro toto *Williams, Sam*: Free as in Freedom. Richard Stallman’s Crusade for Free Software; Beijing [... etc.]: O’Reilly, 2002, ISBN 0-596-00287-4, passim. For a broader survey cf. pars pro toto *Moody*: Die Software-Rebellen, 2001, passim. A very short version is delivered by Richard M. Stallman himself where he states that – in the years while the early free community were destroyed – he saw the “nondisclosure agreement” which must be signed, “[...] even to get an executable copy” as a clear “[...] promise not to help your neighbour”: “A cooperating community was forbidden.” (cf. *Stallman, Richard M.*: The GNU Project; originally published in ‘Open Sources: Voices from the Open Source Revolution, O’Reilly, 1999’; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, p. 16).

⁵²) cf. *Stallman, Richard M.*: What is Copyleft? originally written in 1996; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, p. 89.

⁵³) Based on the American legal copyright system, this method uses two steps: firstly one states, “[...] that it is copyrighted [...]” and secondly one adds those “[...] distribution terms, which are a legal instrument that gives everyone the rights to use, modify, and redistribute the program’s code or any program derived from it but only if the distribution terms are unchanged” (cf. id., ibid.).

⁵⁴) cf. id., ibid.

⁵⁵) And indeed, in the experience of the authors – sometimes – such simplifications gain their

2 Open Source: The Same Idea, Different Licenses

antinomy of *copyleft licenses* and *permissive licenses* is too misleading for taking it as a serious generalization:

On the one hand, even the 'strongly copylefted' GPL requires also other tasks than just the republishing of derivative works. For example, it also calls for to "[...] give any other recipients of the [GPL licensed] Program a copy of this License along with the Program"⁵⁶. Furthermore, the 'weakly copylefted' licenses require also more and different criteria which has to be fulfilled for acting according to these licenses. For example, the EUPL requires that the licensor who does not directly deliver the binaries together with the sourcecode, must offer a sourcecode version of his work free of charge⁵⁷, while the MPL requires that under the same circumstances a recipient "[...] can obtain a copy of such Source Code Form [...] at a charge no more than the cost of distribution to the recipient [...]"⁵⁸. And last but not least, also the *permissive licenses* require tasks which must be fulfilled for a license compliant usage – moreover, they also require different things. For example, the BSD demands that "the (re)distributions [...] must (retain [and/or]) reproduce the above copyright notice [...]". Because of the structure of the "copyright notice", this compulsory notice implies that the authors / copyright holders of the software must be publicly named⁵⁹. As opposed to this, the Apache License requires that "if the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file" which often means that you have to present central parts of such file publicly⁶⁰ – parts which can contain much more information than only the

independent existence and determine decisions on the management level. But that is not the fault of the managers. It is their job, to aggregate, generalize and simplify information. It is the job of the experts, to offer better viewpoints without overwhelming the others with details.

⁵⁶) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp. §1.

⁵⁷) The German version of the EUPL uses the phrase "problemlos und unentgeltlich(sic!) auf den Quellcode (zugreifen können)" (cf. *Europäische Gemeinschaft a. European commission Joinup*: Open-Source-Lizenz für die Europäische Union; 2007 (URL: <http://joinup.ec.europa.eu/system/files/DE/EUPL%20v.1.1%20-%20Lizenz.pdf>) – reference download: 2013-02-08, pp.3, section 3) while the English version contains the specification "the Source Code is easily and freely accessible" (cf. *European Community a. European commission Joinup*: European Union Public Licence v. 1.1. 2007 (URL: <http://joinup.ec.europa.eu/system/files/EN/EUPL%20v.1.1%20-%20Licence.pdf>) – reference download: 2013-02-08, pp.2, section 3)

⁵⁸) cf. *Open Source Initiative*: Mozilla Public License 2.0 (MPL-2.0); 2013 [n.y.] (URL: <http://opensource.org/licenses/MPL-2.0>) – reference download: 2013-02-07, section 3.2.a.

⁵⁹) cf. *Open Source Initiative*: The BSD 2-Clause License; 2012 [n.y.] (URL: <http://www.opensource.org/licenses/BSD-2-Clause>) – reference download: 2012-07-03, wp.

⁶⁰) cf. *Open Source Initiative*: Apache License, Version 2.0; 2004 [n.y. of the page itself] (URL: <http://opensource.org/licenses/Apache-2.0>) – reference download: 2013-02-07, wp. section 4.4.

2 Open Source: The Same Idea, Different Licenses

names of the authors / copyright holders.

So, no doubt – and contrary to the intuitive interpretation of this taxonomy – each *open source license* must be fulfilled by some actions, even the most permissive one. And for ascertaining these tasks, one has to look into these licenses themselves, not the generalized concepts of licenses taxonomies. Hence again, we have to state that even this well known type of grouping of *open source licenses* does not allow to derive a specific license compliant behavior: The taxonomy might be appropriate, if one wants to live with the implicate messages and generalizations of some of its concepts. But the taxonomy is not an adequate tool to determine, what one has to do for fulfilling an *open source license*. A license compliant behaviour for obtaining the right to use a specific piece of *open source software* must be based on the concrete *open source license* by which the licensor has licensed the software. There is no shortcut.

Nevertheless, human beings need generalizing and structuring viewpoints for enabling themselves to talk about a domain – even if they finally have to regard the single objects of the domain for specific purposes. We think that there is a subtler method to regard and to structure the domain of *open source licenses*. So, we want to offer this other possibility to cluster the *open source licenses*⁶¹:

We think that, in general, licenses have a common purpose: they should protect someone or something against something. The structure of this task is based on the nature of the word 'protect' which is a trivalent verb: it links someone or something who protects, to someone or something who is protected and both combined to something against which the protector protects and against the other one is protected. Licenses in general do that. Therefore, it is also the purpose of open source licenses to protect: They can protect the user (recipient) of the software, its contributor resp. developer and/or distributor, and the software itself. And they can protect them against different threats:

- First, we assume, that - in the context of open source software - the user can be protected against the loss of the right to use it, to modify it, and to redistribute it. Additionally, he can be protected against patent disputes.
- Second, we assume, that open source contributors and distributors can be protected against the loss of feedback in the form of code improvements and derivatives, against warranty claims, and against patent disputes.
- Third, we assume, that the open source programs and their specific forms – may they be distributed or not, may they be modified or not, may they be distributed as binaries or as sources – can be protected against the re-closing resp. against the re-privatizing of their further development.
- Fourth, we want to assume that new on-top developments being based

⁶¹) even if also we have to concede that, ultimately, one has to always look into the license itself

on open source components can be protected against the privatizing for enlarging the world of freely usable software⁶².

With respect to these viewpoints, one gets a subtler picture of the license specific protecting power. Thus, we are going to describe and deduce the protecting power of each of the open source licenses on the following pages. Table 2.1 summarizes the results as a quick reference⁶³.

2.1 The protecting power of the Affero Gnu Public License (AGPL) [tbd]

- The Affero Gnu Public License protects ...
- But the Affero Gnu Public License does not protect ...

2.2 The protecting power of the Apache License (ApL)

As an approved *open source license*⁶⁴, the Apache License⁶⁵ protects the user against the loss of the right to use, to modify and/or to distribute the received copy of the source code or the binaries⁶⁶. Furthermore, based on its patent clause⁶⁷, the ApL protects the users against patent disputes⁶⁸. Because of this patent clause and of its “disclaimer of warranty” together with its “limitation of liability”, the Apache license also protects the contributors / distributors against

⁶²) In a more rigid version, this capability of a license could also be identified as the power to protect the community against a stagnation of the set of open source software – but this description is at least a little too long to be used by the following pages

⁶³) → table 2.1 on p. 24

⁶⁴) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.

⁶⁵) The Apache License, version 2.0 is maintained by the Apache Software Foundation (cf. *Apache Software Foundation*: Apache License, Version 2.0; 2004 [URL: <http://www.apache.org/licenses/LICENSE-2.0>] – reference download: 2011-08-31, wp). Of course, also the OSI is hosting a duplicate of the Apache license (cf. *Open Source Initiative*: APL-2.0, 2004, wp.) and is listing it as an officially approved open source license (cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.). The Apache license 1.1 is classified by the OSI as “superseded license” (cf. *Open Source Initiative*: The Open Source Licenses by Category, 2013, wp.). In the same spirit, also the Apache Software Foundation itself classifies the releases 1.0 and 1.1 as “historic” (cf. *Apache Software Foundation*: Licenses; 2013 [n.y.] [URL: <http://www.apache.org/licenses/>] – reference download: 2013-02-25, wp.). Thus, the OSLiC only focuses on the most recent APL-2.0 version. For those, who have to fulfill these earlier Apache licenses it could be helpful to read them as siblings of the BSD-2CL and BSD-3CL licenses.

⁶⁶) cf. *Open Source Initiative*: APL-2.0, 2004, wp §2.

⁶⁷) → OSLiC pp. 42

⁶⁸) cf. id., l.c., wp §3.

2 Open Source: The Same Idea, Different Licenses

Table 2.1: Open Source Licenses as Protectors

Open Source Licenses ^a		are protecting												
		Users			Contributors (Distributors)			Open Source Software						On-Top Develop.
								not distrib- uted	distributed as					
		unmodified sources		modified binaries		Privatizings								
who have already got sources or binaries			who spread open source software				against							
							the loss of the right to			Re-Closings / Re-Privatizings of already opened software				
use it	modify it	redistribute it	Patent Disputes	Loss of Feedback	Warranty Claims	Patent Disputes								
ApL	2.0	✓	✓	✓	✓	┐	✓	✓	┐	✓	┐	✓	┐	┐
BSD	3-Cl	✓	✓	✓	┐	┐	✓	┐	┐	✓	┐	✓	┐	┐
	2-Cl	✓	✓	✓	┐	┐	✓	┐	┐	✓	┐	✓	┐	┐
MIT		✓	✓	✓	┐	┐	✓	┐	┐	✓	┐	✓	┐	┐
MS-PL		✓	✓	✓	✓	┐	✓	✓	┐	✓	┐	✓	┐	┐
PgL		✓	✓	✓	┐	┐	✓	┐	┐	✓	┐	✓	┐	┐
PHP	3.0	✓	✓	✓	┐	┐	✓	┐	┐	✓	┐	✓	┐	┐
CDDL	1.0	✓	✓	✓	—	—	—	—	—	—	—	—	—	—
EPL	1.0	✓	✓	✓	✓	✓	✓	✓	┐	✓	✓	✓	✓	┐
EUPL	1.1	✓	✓	✓	✓	✓	✓	✓	┐	✓	✓	✓	✓	┐
LGPL	2.1	✓	✓	✓	┐	✓	✓	┐	┐	✓	✓	✓	✓	┐
	3.0	✓	✓	✓	✓	✓	✓	✓	┐	✓	✓	✓	✓	┐
MPL	2.0	✓	✓	✓	✓	✓	✓	✓	┐	✓	✓	✓	✓	┐
MS-RL		✓	✓	✓	—	—	—	—	—	—	—	—	—	—
AGPL	3.0	✓	✓	✓	—	—	—	—	—	—	—	—	—	—
GPL	2.1	✓	✓	✓	—	—	—	—	—	—	—	—	—	—
	3.0	✓	✓	✓	—	—	—	—	—	—	—	—	—	—

^{a)} '✓' indicates that the license protects with respect to the meaning of the column, '¬' indicates that the license does not protect with regard to the meaning of the column, and '-' indicates, that the corresponding statement must still be evaluated. *Slanted names of licenses* indicate that these licenses are only listed in this table while the corresponding mindmap (→ p. 36) does not cover them

2 Open Source: The Same Idea, Different Licenses

patents disputes and warranty claims⁶⁹. Finally, the ApL protects the distributed sources themselves *against* a change of the license which would *reset* the work *as closed software*, because first, one “[...] must give any other recipients of the work or derivative works a copy of (the Apache) license”, second, “in the source form of any derivative works that (one) distributes”, one has “[...] to retain [...] all copyright, patent, trademark, and attribution notices [...]”, and third, one must “[...] include a readable copy [...] of the] NOTICE file” being supplied by the original package one has received⁷⁰.

But the Apache License does not protect the contributors against the loss of feedback because it does not ‘copyleft’ the software: the Apache license does not contain any sentence requiring that one has also to publish the source code. In the same spirit, the ApL does not protect the undistributed software or the distributed binaries against re-closings – neither in unmodified nor in modified form – because the Apache License allows to (re)distribute the binaries without also supplying the sources – even if the binaries rest upon sources modified by the distributor. Finally, the ApL does not protect the on-top developments against a privatizing.

2.3 The protecting power of the BSD licenses

As approved *open source licenses*⁷¹, the BSD Licenses⁷² protect the user against the loss of the right to use, to modify and/or to distribute the received copy of the source code or the binaries⁷³. Additionally, they protect the contributors and/or distributors against warranty claims of the software users, because these licenses contain a ‘No Warranty Clause’⁷⁴. And finally they protect the distributed sources against a change of the license which closes the sources, because each modification and “redistributions of [the] source code must retain the [...] copyright notice, this list of conditions and the [...] disclaimer”⁷⁵: Therefore it is incorrect to distribute a BSD licensed code under another license – regardless, whether it closes the sources or not⁷⁶.

⁶⁹) cf. *Open Source Initiative*: APL-2.0, 2004, wp §3, §7, §8.

⁷⁰) cf. *id.*, l.c., wp §4.

⁷¹) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.

⁷²) BSD has to be resolved as *Berkely Software Distribution*. For details of the BSD license release and namings cf. *Open Source Initiative*: The BSD 3-Clause License; 2012 [n.y.] (URL: <http://www.opensource.org/licenses/BSD-3-Clause>) – reference download: 2012-07-04, wp. editorial

⁷³) cf. *Open Source Initiative*: The Open Source Definition, 2012, wp §1ff.

⁷⁴) one for all version cf. *Open Source Initiative*: The BSD 2-Clause License, 2012, wp.

⁷⁵) cf. *id.*, *ibid.*

⁷⁶) In common sense based discussions you may have heard that BSD licenses allow to republish the work under another, an own license. Taking the words of the BSD License seriously that

But the BSD Licenses protect neither the users nor the contributors and/or distributors against patent disputes (because they do not contain any patent clause). They do not protect the contributors against the loss of feedback (because they do not 'copyleft' the software). Moreover, they do not protect the undistributed software or the distributed binaries against re-closings – neither in unmodified nor in modified form – because they allow to redistribute only the binaries without also supplying the source code⁷⁷. Finally, the BSD licenses do not protect the on-top developments against a privatizing.

2.4 The protecting power of the Eclipse Public License (EPL)

As an approved *open source license*⁷⁸, the Eclipse Public License⁷⁹ protects the user against the loss of the right to use, to modify and/or to distribute the received copy of the source code or the binaries⁸⁰. Furthermore, based on its patent clause⁸¹, the EPL protects the users also against patent disputes⁸². Besides this patent clause, the EPL contains the sections “no warranty” and “disclaimer of liability”⁸³. These three elements together protect the contributors / distributors against patents disputes and warranty claims. Finally, the EPL protects the distributed sources themselves *against* a change of the license which would *reset* the work *as closed software*: First, the Eclipse Public Licenses requires that if a work – released under the EPL – “[...] is made available in source code form [...] (then) it must be made available under this (EPL) agreement, too” while this act of ‘making available’ “must” incorporate a “copy” of the EPL into “each

is not valid under all circumstances: Yes, it is true, you are not required to redistribute the sourcecode of a modified (derivative) work. You are allowed to modify a received version and to distribute the results only as binary code and to keep your improvements closed. But if you distribute the source code of your modifications, you have retain the licensing, because “Redistribution [...] in source [...], with or without modification, are permitted provided that [...] (the) redistributions of source code [...] retain the above copyright notice, this list of conditions and the following disclaimer” (cf. *Open Source Initiative*: The BSD 2-Clause License, 2012, wp)

⁷⁷) see both, the BSD-2CL License (cf. id., ibid.), and the BSD-3CL License (cf. *Open Source Initiative*: The BSD 3-Clause License, 2012, wp)

⁷⁸) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.

⁷⁹) The Eclipse Public License, version 1.0 is maintained by the Eclipse Software Foundation (cf. *Eclipse Foundation*: Eclipse Public License, Version 1.0; 2005 [n.y. of the page itself] (URL: <http://www.eclipse.org/org/documents/epl-v10.php>) – reference download: 2013-02-20, wp). Of course, also the OSI is hosting a duplicate (cf. *Open Source Initiative*: Eclipse Public License, Version 1.0; 2005 [n.y. of the page itself] (URL: <http://opensource.org/licenses/EPL-1.0>) – reference download: 2013-02-20, wp).

⁸⁰) cf. id., l.c., wp §2a.

⁸¹) → OSLiC pp. 43

⁸²) cf. id., l.c., wp §2b & §2c.

⁸³) cf. id., l.c., wp §5 & §6.

2 Open Source: The Same Idea, Different Licenses

copy of the [distributed] program” or program package⁸⁴. But in opposite to the permissive licenses, the EPL does not only protect the distributed source code – regardless whether it is modified or not. The EPL also protects the distributed modified or unmodified binaries: The EPL allows each modifying “contributor” and distributor “[...] to distribute the Program in object code form under (one’s) own license agreement [...]” provided this license clearly states that the “source code for the Program is available” and where the “licensees” can “[...] obtain it in a reasonable manner on or through a medium customarily used for software exchange”⁸⁵. Thus, one has to conclude that the EPL is a copyleft license.

But the Eclipse Public License is not a license with strong copyleft; the EPL uses ‘only’ a weak copyleft effect⁸⁶: Indeed, the EPL says that for each EPL licensed “program” – distributed in object form – a place must be made known where one can get the corresponding source code⁸⁷. The term ‘Program’ is defined as any “Contribution distributed in accordance with [...] (the EPL)” while the term ‘Contribution’ refers - besides other elements - to “changes to the Program, and additions to the Program”⁸⁸. Unfortunately, this is a circular definition: ‘Program’ is defined by ‘Contribution’; and ‘Contribution’ is defined by ‘Program’. Nevertheless, one has to read the license benevolently. Uncontroversial should be this: If one distributes any modified EPL licensed program, library, module, or plugin, then one has to publish the modified source code, too. If one “adds” some own plugins or additional libraries which are used by an EPL licensed program (which on behalf of this use must have been modified by adding [sic!] procedure calls) then one has publish the code of both parts: that of the program and that of the added elements. In this sense, the EPL clearly protects the binaries against re-closings like other weak copyleft using licenses. But if one distributes only an EPL licensed library which is used as a component by another not EPL licensed on-top program, then this library does not depend on the top development – provided that the library itself does not call any (program) functions or procedures delivered by the overarching on-top development. Hence, nothing is added to the library; and hence, no other code than that of the library must be published. Therefore, the EPL does not use the strong copyleft effect in the meaning of the GPL.

⁸⁴) cf. *Open Source Initiative*: EPL-1.0, 2005, wp §3.

⁸⁵) cf. id., l.c., wp. §3, esp. §3.b.iv.

⁸⁶) Even if one can find contrary specifications in the internet. Pars pro toto cf. *ifross*: ifross Lizenz-Center, 2011, wp.: This page is listing the EPL in the section “Other Licenses with strong Copyleft Effect”

⁸⁷) cf. *Open Source Initiative*: EPL-1.0, 2005, wp. §3, esp. §3.b.iv.

⁸⁸) cf. id., l.c., wp. §1.

2.5 The protecting power of the European Union Public License (EURL)

As an approved *open source license*⁸⁹, the European Union Public License⁹⁰ protects the user against the loss of the right to use, to modify and/or to distribute the received copy of the source code or the binaries⁹¹. Furthermore, based on its patent clause⁹², the EURL protects the users against patent disputes⁹³. Besides this patent clause, the EURL additionally contains a “Disclaimer of Warranty” and a “Disclaimer of Liability”⁹⁴. These three elements together protect the contributors / distributors against patents disputes and warranty claims. Finally, the EURL also protects the distributed sources against a re-closing / re-privatizing and the contributors against the loss of feedback. This protection is based on two steps: First, the European Public License contains a particular paragraph titled “Copyleft clause” which stipulates that “copies of the Original Work or Derivative Works based upon the Original Work” must be distributed “under the terms of (the European Union Public) License”⁹⁵. Second, the EURL requires that each licensee – as long as he “[...] continues to distribute and/or communicate the Work” – has also to “[...] provide [...] the Source Code”, either directly or by “[...] (indicating) a repository where this Source will be easily and freely available [...]”⁹⁶. This condition seems to be so important for the EURL that the license repeats its message: in another paragraph the EURL requires again that “if the Work is provided as Executable Code, the Licensor provides in addition a machine-readable copy of the Source Code of the Work along with each copy of the Work [...] or indicates, in a notice [...], a repository where the Source Code is easily and freely accessible for as long as the Licensor continues to dis-

⁸⁹) cf. *Open Source Initiative: The Open Source Licenses*, alphabetically sorted, 2012, wp.

⁹⁰) The European Union Public License, version 1.1 is maintained by the European Union and hosted under the label “Joinup” (cf. *European Community a. European commission Joinup: EURL-1.1/EN*, 2007, wp). This EURL has officially been translated into many languages, among others into German (cf. *Europäische Gemeinschaft a. European commission Joinup: EURL-1.1/DE*, 2007, wp). Because of this multi lingual instances, the OSI does not offer its own version, but just a landing page linked to the landing page of the European host “Joinup” (cf. *Open Source Initiative: European Union Public License, version 1.1 (EURL-1.1; 2007 [n.y. of the html page itself]* [URL: <http://opensource.org/licenses/EURL-1.1>] – reference download: 2013-03-04, wp).

⁹¹) cf. *Europäische Gemeinschaft a. European commission Joinup: EURL-1.1/DE*, 2007, wp. §2.

⁹²) → OSLiC pp. 44

⁹³) cf. *European Community a. European commission Joinup: EURL-1.1/EN*, 2007, wp. §2, at its tail.

⁹⁴) cf. id., l.c., wp. §7 & §8.

⁹⁵) cf. id., l.c., wp. §5.

⁹⁶) cf. id., ibid.

2 Open Source: The Same Idea, Different Licenses

tribute [...] the Work”⁹⁷. Based on the meaning of “Work” which is defined by the EUPL as “the Original Work and/or its Derivative Works”⁹⁸ it must be concluded that the EUPL is a copyleft license.

But nevertheless, the European Union Public License is not a license with strong copyleft: On the one hand, if one takes the core of the EUPL then the license seems to protect not only the modifications of the original work against re-closings and (re-)privatizings, but also the on-top developments because normally you have to publish the source code in both cases. Understood in this way, the EUPL would be a ‘strong copyleft license’. But on the other hand, the EUPL additionally contains a “Compatibility clause” stating that “if the Licensee Distributes [...] Derivative Works or copies thereof based upon both the Original Work and another work licensed under a Compatible Licence, this Distribution [...] can be done under the terms of this Compatible Licence”⁹⁹ – while the term “Compatible Licence” is explicitly defined by a list of compatible licenses, for example the Eclipse Public License¹⁰⁰. Based on this compatibility clause the obligation to publish the code of an on-top development can be subverted: As first step, you could release a little, more or less futile on-top application licensed under the Eclipse Public License¹⁰¹ which uses a library licensed under the EUPL. As second step, you add this ‘EUPL library’ which you now may also distribute under the EPL instead of retaining the EUPL licensing. So, finally you obtain the same work under the Eclipse Public License which is a weak copyleft license¹⁰². Hence the protection of the EUPL-1.1 is not as comprehensive as one might assume on the base of the license text itself¹⁰³, it can at most be a weak copyleft license – even if the reader might get the impression that the authors of the EUPL wished

⁹⁷) cf. *European Community a. European commission Joinup*: EUPL-1.1/EN, 2007, wp. §3.

⁹⁸) cf. id., l.c., wp. §1.

⁹⁹) cf. id., l.c., wp. §5.

¹⁰⁰) cf. id., l.c., wp. Appendix.

¹⁰¹) Taking the license text very seriously, it is not even necessary that this little futile application must depend on the EUPL library by calling functions of EUPL library. The license text only says that “another [any other] work licensed under a Compatible Licence” can be distributed together with “derivative works”. By this wording, the license itself is establishing a contrast between the derivative work and the other work - what indicates that the other work has not necessarily also to be a derivative work.

¹⁰²) → OSLiC, p. 26

¹⁰³) This kind of specifying the protective power of the EUPL is initially be presented by the FSF (cf. *Free Software Foundation: Various Licenses and Comments about Them* [; 2013 [n.y.] (URL: <http://www.gnu.org/licenses/license-list.en.html#EUP>) – reference download: 2013-03-05, pp.wp. section ‘European Union Public License’). The EU answers that publishing such a trick will comprise its user in the eyes of the open source community (cf. *European Community a. European commission Joinup*: New FSF statements on the EUPL are a step in the right direction; 2013 [n.y.] (URL: <https://joinup.ec.europa.eu/community/eupl/news/new-fsf-statements-eupl-are-step-right-direction>) – reference download: 2013-03-05, p.wp.). That is undoubtedly true. But unfortunately, this argument does not close the hole in the protecting shield put up by the EUPL

to write a strong copyleft license. Howsoever, the EUPL license does not protect the on-top developments against a privatizing.

2.6 The protecting power of the Gnu Public License (GPL) [tbd]

The Gnu Public License protects ...

But the Gnu Public License does not protect ...

2.7 The protecting power of the Lesser Gnu Public License (LGPL) [tbd]

The Lesser Gnu Public License protects ...

But the Lesser Gnu Public License does not protect ...

2.8 The protecting power of the MIT license

As an approved *open source license*¹⁰⁴, the MIT License¹⁰⁵ protects the user against the loss of the right to use, to modify and/or to distribute the received copy of the source code or the binaries¹⁰⁶. Additionally, it protects the contributors and/or distributors against warranty claims of the software users, because it contains a 'No Warranty Clause'¹⁰⁷. And finally it protects the distributed sources against a change of the license which would close the sources, because the "permission [...] to use, copy, modify, [...] distribute, [...] (is granted) subject to the [...] conditions, [that] the [...] copyright notice and this permission notice shall be included in all copies or substantial portions of the Software"¹⁰⁸

¹⁰⁴) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.

¹⁰⁵) MIT has to be resolved as "Massachusetts Institute of Technology" (cf. *Wikipedia (en)*: MIT License; n.l., 2011 ⟨URL: http://en.wikipedia.org/wiki/MIT_License⟩ – reference download: 2011-09-20, wp.).

¹⁰⁶) cf. *Open Source Initiative*: The Open Source Definition, 2012, wp 1ff.

¹⁰⁷) cf. *Open Source Initiative*: The MIT License; 2012 [n.y.] ⟨URL: <http://opensource.org/licenses/mit-license.php>⟩ – reference download: 2012-08-24, wp.

¹⁰⁸) cf. id., *ibid.*. The argumentation why the source code is protected, but not the binary form follows that of the BSD licenses: By these requirements, one is not obliged to redistribute the sourcecode of a modified (derivative) work. One is allowed to modify a received version and to distribute the results only in binary form and to keep one's improvements closed. But if one distribute the source code of the modifications, the licensing is retained, simply because the MIT "[...] permission note shall be included in all copies or substantial portions of the software".

But the MIT License does not protect the users or the contributors and/or distributors against patent disputes (because it does not contain any patent clause). Additionally, it does not protect the contributors against the loss of feedback (because it does not 'copyleft' the software). Moreover, the MIT license does not protect the undistributed software or the distributed binaries against re-closings – neither in unmodified nor in modified form – because it allows to redistribute only the binaries without also supplying the source code¹⁰⁹. Finally, the MIT license does not protect the on-top developments against a privatizing.

2.9 The protecting power of the Mozilla Public License (MPL)

As an approved *open source license*¹¹⁰, the Mozilla Public License¹¹¹ protects the user against the loss of the right to use, to modify and/or to distribute the received copy of the source code or the binaries¹¹². Furthermore, based on its split and distributed patent clause¹¹³, the MPL protects the users against patent disputes¹¹⁴. Besides this patent sections, the MPL additionally contains a “Disclaimer of Warranty” and a “Limitation of Liability”¹¹⁵. These three elements together protect the contributors / distributors against patents disputes and war-

¹⁰⁹) cf. *Open Source Initiative*: The MIT License, 2012, wp.

¹¹⁰) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.

¹¹¹) In 2012, the Mozilla Public License 2.0 (cf. *Mozilla Foundation*: Mozilla Public License 2.0 (MPL-2.0); 2012 [URL: <http://www.mozilla.org/MPL/2.0/>] – reference download: 2013-03-05, wp.) has been released as a result of a longer “Revision Process” (cf. *Mozilla Foundation*: About MPL 2.0: Revision Process and Changes FAQ; 2013 [n.y.] [URL: <http://www.mozilla.org/MPL/1.1/>] – reference download: 2013-03-05, wp.) by which the Mozilla Public License 1.1 (cf. *Mozilla Foundation*: Mozilla Public License Version 1.1; 2013 [n.y.] [URL: <http://www.mozilla.org/MPL/1.1/>] – reference download: 2013-03-05, wp.) has been ousted. The OSI is also hosting its version of the MPL-2.0 (cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp.) and is listing it as an OSI approved license (cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.) while it classifies the MPL-1.1 as a “superseded license” (cf. *Open Source Initiative*: The Open Source Licenses by Category, 2013, wp.). The Mozilla Foundation itself says concerning the difference between the two licenses that “the most important part of the license - the file-level copyleft - is essentially the same in MPL 2.0 and MPL 1.1” (cf. *Mozilla Foundation*: MPL 2.0: Revision Process and Changes, 2013, wp.). By reading the MPL-1.1, one could get the impression that fulfilling all conditions of the MPL-2.0 would imply also to act in accordance to the MPL-1.1. Thus the OSLiC focuses on the MPL-2.0, at least for the moment. Nevertheless, in this section we want to use the general label ‘MPL’ without any releasenummer for indicating that with respect to its protecting power the MPL-2.0 and the MPL-1.1 can be taken as equipollent.

¹¹²) cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp. §2.1.a.

¹¹³) → OSLiC pp. 47

¹¹⁴) cf. id., l.c., wp. §2.1.b, §2.3, §5.2.

¹¹⁵) cf. id., l.c., wp. §6 & §7.

2 Open Source: The Same Idea, Different Licenses

ranty claims. Finally, the MPL also protects the distributed sources against a re-closing / re-privatizing and the contributors against the loss of feedback: The MPL clearly says that, on the one hand, “all distribution of Covered Software in Source Code Form, including any Modifications[...] must be under the terms of this License”¹¹⁶ and that, on the other hand, an MPL licensed software “[...] (distributed) in Executable Form [...] must also be made available in Source Code Form [...]”¹¹⁷. So, it must be inferred that the MPL is a copyleft license.

But nevertheless, the Mozilla Public License is not a license with strong copyleft. It does not protect on-top developments against privatizings: First, the MPL does not use the term *derivative work*¹¹⁸. Instead of this, the MPL denotes the “[...] (initial) Source Code Form [...] and Modifications of such Source Code Form” by the label “Covered Software”¹¹⁹ – while the term “Modifications” refers to “any file in Source Code Form that results from an addition to, deletion from, or modification of the contents of Covered Software or any file in Source Code Form that results from an addition to, deletion from, or modification of the contents of Covered Software”¹²⁰. Second, the MPL contrasts the source code form and its modifications with the “Larger Work” by specifying that the larger work is “[...] material, in a separate file or files, that is not covered software”¹²¹. Finally, the MPL states, that “you may create and distribute a Larger Work under terms of Your choice, provided that You also comply with the requirements of this License for the Covered Software”¹²². Based on these specifications, one has to reason that an on-top development which depends on MPL licensed libraries by calling some of their functions, is undoubtably a derivative work¹²³, but also only a larger work in the meaning of the MPL so that code of this on-top application needs not to be published – provided, that the library and the on-top development are distributed as different files¹²⁴. Hence, the MPL is license with a weak copyleft

¹¹⁶⁾ cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp. §3.1.

¹¹⁷⁾ cf. *id.*, l.c., wp. §3.2.

¹¹⁸⁾ cf. *id.*, l.c., wp. The MPL-1.1 uses the term *derivative work* only in the context of writing new “versions of the license”, not in the context of licensing software (cf. *Mozilla Foundation*: Mozilla Public License Version 1.1, 2013, wp. §6.3).

¹¹⁹⁾ cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp. §1.4.

¹²⁰⁾ cf. *id.*, l.c., wp. §1.10. The Mozilla Foundation denotes this reading by the term “file-level copyleft” (cf. *Mozilla Foundation*: MPL 2.0: Revision Process and Changes, 2013, wp).

¹²¹⁾ cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp. §1.7.

¹²²⁾ cf. *id.*, l.c., wp. §3.3.

¹²³⁾ This follows from the general meaning of a *derivative work* as a benevolent software developer would read this term (\rightarrow OSLiC, pp. 49). But again: The MPL does not focus on this general aspect; it uses its own concept of a *larger work*.

¹²⁴⁾ It might be discussed whether integrating a declaration of a function, class, or method into the on-top development by including the corresponding header files indeed means that one is “including portions (of the Source Code Form)” into a file which therefore has to be taken as “Modification” (cf. *Mozilla Foundation*: Mozilla Public License Version 1.1, 2013, wp. §1.4). From the viewpoint of a benevolent developer it should be difficult to argue that the

effect and does not protect the on-top developments against privatizings.

2.10 The protecting power of the Microsoft Public License (MS-PL)

As an approved *open source license*¹²⁵, the Microsoft Public License protects the user against the loss of the right to use, to modify and/or to distribute the received copy of the source code or the binaries¹²⁶. Furthermore, based on its patent clause¹²⁷, the MS-PL protects the users against patent disputes¹²⁸. Because of this patent clause and of its concise *disclaimer of warranty*, the MS-PL also protects the contributors / distributors against patents disputes and warranty claims¹²⁹. Finally, the Microsoft Public License protects the distributed sources themselves - and even “portions of these sources” – *against* a change of the license which would *reset* the work *as closed software*, because first, one “[...] must retain all copyright, patent, trademark, and attribution notices that are present in the software”¹³⁰, and because second, one must also incorporate “a complete copy of this license” into one’s own distribution premised one distributes the source code¹³¹.

But the Microsoft Public License does not protect the contributors against the loss of feedback because it does not ‘copyleft’ the software: The license does not contain any sentence which requires that one has to publish the sources, too¹³².

including of declaring (header) files alone can evoke a derivative work. It is the call of the function in one’s code which establishes the dependency. But that is not the point, the MPL focuses. The MPL aims on the textual reuse of (defining) code snippets. Hence, one could ignore the textual integration of parts of the declaring header files: it should not trigger that one’s own work becomes a modification in the eyes of the Mozilla Foundation. But of course, one would circumvent the idea of the MPL if one hides defining code in header files and reuses that code by one’s own compilation. This would undoubtedly be an incorporation of portions and therefore would make the incorporating file becoming a modification of the MPL licensed initial work.

¹²⁵) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.

¹²⁶) cf. *Open Source Initiative*: Microsoft Public License (MS-PL); 2013 [n.y.] (URL: <http://opensource.org/licenses/MS-PL>) – reference download: 2013-02-26, wp. §2.

¹²⁷) → OSLiC pp. 48

¹²⁸) cf. id., l.c., wp. §2.B and §3.B.

¹²⁹) cf. id., l.c., wp. §2B, §3B, §3D.

¹³⁰) cf. id., l.c., wp. §3C.

¹³¹) cf. id., l.c., wp. §3D.

¹³²) There seems to be some misunderstandings on the internet: The English wikipedia specifies the MS-PL as a permissive license and the MS-RL as a license with copyleft effect (cf. *Wikipedia (en)*: Shared source; n.l, 2013 [n.y.] (URL: http://en.wikipedia.org/wiki/Shared_source) – reference download: 2013-02-26, wp.). The German wikipedia says that the MS-PL is a license with a “schwachen [weak] copyleft” (cf. *Wikipedia (de)*: Microsoft Public License; n.l, 2013 [n.y.] (URL: http://de.wikipedia.org/wiki/Microsoft_Public_License)).

In the same spirit, the MS-PL does not protect the undistributed software or the distributed binaries against re-closings – neither in unmodified nor in modified form – because the MS-PL License allows to (re)distribute the binaries without also supplying the sources – even if the binaries rest upon sources modified by the distributor. Finally, also the MS-PL does not protect the on-top developments against a privatizing.

2.11 The protecting power of the Postgres License (PgL)

As an approved *open source license*¹³³, the PostgreSQL License protects the user against the loss of the right to use, to modify and/or to distribute the received copy of the source code or the binaries¹³⁴. Because of its *disclaimer of warranty*, the PgL also protects the contributors / distributors against warranty claims¹³⁵. Finally, the PgL protects the distributed sources themselves *against* a change of the license which would *reset* the work *as closed software*, because the “copyright notice” and the whole license must “[...] appear in all copies”¹³⁶.

But the PostgreSQL License does not protect the contributors against the loss of feedback because it does not ‘copyleft’ the software: The license does not contain any sentence which requires that one has to publish the sources, too. In the same spirit, the PgL does not protect the undistributed software or the distributed binaries against re-closings – neither in unmodified nor in modified form – because the PgL allows to (re)distribute the binaries without also supplying the sources – even if the binaries rest upon sources modified by the distributor. Finally, the PgL does not protect the on-top developments against a privatizing.

License} – reference download: 2013-02-26, wp.). And it says also that the “Microsoft Reciprocal License” (MS-RL) is a license with weak copyleft, too (cf. *Wikipedia (de)*: Microsoft Reciprocal License; n.l., 2013 [n.y.] (URL: <http://de.wikipedia.org/wiki/Ms-RL>) – reference download: 2013-02-26, wp.). But for the very thoroughly working “ifross license center”, the MS-RL is a license with restricted (weak) copyleft, while the MS-PL is a permissive license with some selectable options (cf. *ifross*: ifross Lizenz-Center, 2011, wp.). Based on the license text itself and these other readings, we decided to take the MS-PL as a permissive license in accordance to the English wikipedia page and the ifross page.

¹³³) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.

¹³⁴) cf. *Open Source Initiative*: The PostgreSQL Licence (PostgreSQL); 2013 [n.y.] (URL: <http://opensource.org/licenses/PostgreSQL>) – reference download: 2013-02-27, wp..

¹³⁵) cf. id., ibid.

¹³⁶) cf. id., ibid.

2.12 The protecting power of the PHP License

As an approved *open source license*¹³⁷, the PHP-3.0 License protects the user against the loss of the right to use, to modify and/or to distribute the received copy of the source code or the binaries¹³⁸. Because of its *disclaimer of warranty*, the PHP license also protects the contributors / distributors against warranty claims¹³⁹. Finally, the PHP license protects the distributed sources themselves *against* a change of the license which would *reset* the work *as closed software*, because “redistributions of source code must retain the [...] copyright notice, this list of conditions and the [...] disclaimer”¹⁴⁰.

But the PHP-3.0 License does not protect the contributors against the loss of feedback because it does not ‘copyleft’ the software: The license does not contain any sentence which requires that one has to publish the sources, too. In the same spirit, the PHP license does not protect the undistributed software or the distributed binaries against re-closings – neither in unmodified nor in modified form – because the PHP license allows to (re)distribute the binaries without also supplying the sources – even if the binaries rest upon sources modified by the distributor.

2.13 Summary

All these specifications cannot only be summarized by a table¹⁴¹, but also by a mindmap:

¹³⁷) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.

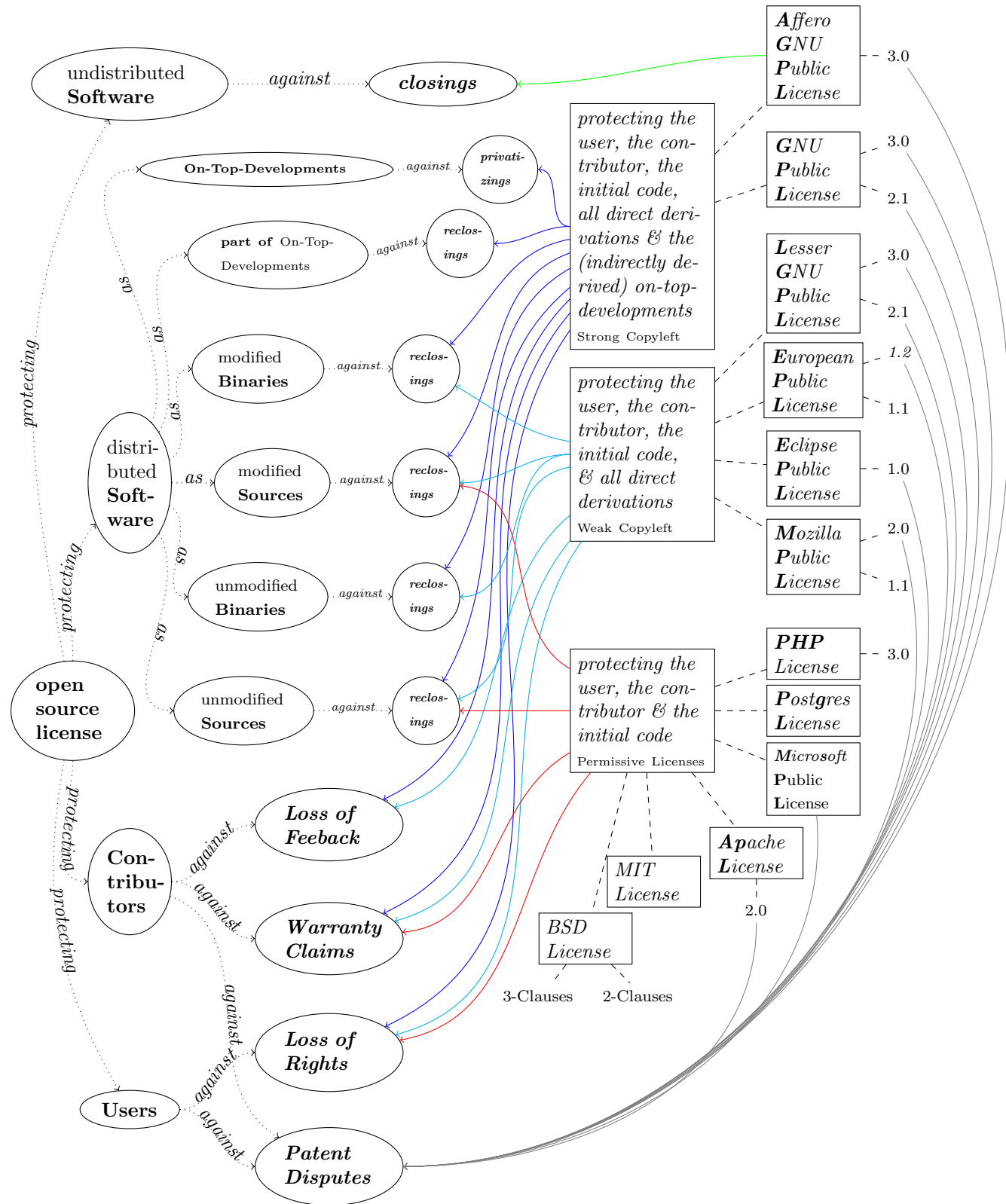
¹³⁸) cf. *Open Source Initiative*: The PHP License 3.0 (PHP-3.0); 2013 [n.y.] (URL: <http://opensource.org/licenses/PHP-3.0>) – reference download: 2013-02-27, wp..

¹³⁹) cf. id., ibid.

¹⁴⁰) cf. id., ibid.

¹⁴¹) → OSLiC, p. 24

2 Open Source: The Same Idea, Different Licenses



Finally, one could generate new groups of open source license, new classes, like

2 Open Source: The Same Idea, Different Licenses

'user protecting licenses'¹⁴², 'patent disputes fending licenses', an so on.

However, it must kept in mind that all of these grouping viewpoints do not permit the conclusion that all members of a group can be respected by fulfilling the same requirements. This would only be possible if the grouping criteria would directly refer to the fulfilling tasks. Indeed, nearly all open source licenses do differ with respect to these criteria, and even if the differences are very small, they can't be neglected¹⁴³. So: reflecting on possible classes of open source licenses is a good method to become familiar with the area of open source licenses. But it is not a method to determine, what one needs to be done to obtain the right to use the software. For that purpose every license must be considered individually.

¹⁴²⁾ all of them because all of them have to fulfill the OSD

¹⁴³⁾ Pars pro toto: Both, the BSD license and the Apache license require that you provide an indication to the developers of the application. But in case of the BSD license you have to publish the copyright notice / line, while in case of the Apache license you have exactly to present the content of the notice file distributed together with the application.

3 Open Source: About Some Side Effects

3.1 The problem of implicitly releasing patents

In this chapter, we are briefly analyzing the effect of patent clauses in open source licenses – not in general, but with respect to the license fulfilling tasks they require, also known as the ‘implicit acceptance of a patent use’ by distributing open source software.

At least the free software movement frowns on the existence of software patents¹⁴⁴. One of the most known witnesses for that attitude is the GPL itself. Its preamble purports that “[...] any free program is threatened constantly by software patents”¹⁴⁵. One can read that the open source community fears three risks: First, they are apprehensive of people who hijack the idea of a piece of open source software they do not have developed, register a corresponding patent, and finally try to earn money by preventing the use of the software or by involving its users into patent litigations¹⁴⁶. Second, they fear a bramble of general software patents which practically prohibits to develop open source software legally¹⁴⁷. Third, they anticipate the possibility that (not quite benevolent) open source

¹⁴⁴) For an early and elaborated description on the effects of software patents based on the viewpoint of the free software movement see *Stallman, Richard M.: Free Software: Freedom and Cooperation*; transcript of a speech given at New York University on 29 May 2001; In *Stallman: Free Software, Free Society: Selected Essays*, 2002, wp. This lecture seems to have been given more than once and printed later on (cf. *Stallman, Richard M.: The Danger of Software Patents*; transcript of a speech given at University of Cambridge, London on the 25th of March 2002; In *Stallman: Free Software, Free Society: Selected Essays*, 2002, wp). Within the first decade of 2000, the focus switched to a more political fight against software patents (cf. *Stallman, Richard M.: Fighting Software Patents - Singly and Together*; n.st. [2004] (URL: <http://www.gnu.org/philosophy/fighting-software-patents.html>) – reference download: 2013-02-18, wp). But recently there seems to have been appeared another turn in dealing with software patents: Not fighting against the patents, but mitigating their effects: The proposal is ‘[...] (to legislate) that developing, distributing, or running a program on generally used computing hardware does not constitute patent infringement’ (cf. *Stallman, Richard M.: Let’s Limit the Effect of Software Patents, Since We Can’t Eliminate Them*; in: *Wired*, n.st. January (2012) (URL: <http://www.wired.com/opinion/2012/11/richard-stallman-software-patents/>) – reference download: 2013-02-18, ISSN n.st., wp)

¹⁴⁵) cf. *Open Source Initiative: The GPL-2.0 License (OSI)*, 1991, p. wp.

¹⁴⁶) cf. *Jaeger a. Metzger: Open Source Software. Rechtliche Rahmenbedingungen der Freien Software*, 2011, p. 234.

¹⁴⁷) cf. id., ibid.

3 Open Source: About Some Side Effects

developers could try to register patents for undermining the open source principles¹⁴⁸.

Howsoever, regardless whether one tries to fight against software patents or not, software patents have come true. To act law-abidingly requires to manage the constraints of patents properly. Open source licenses know and respect this necessity. Moreover, at least some of them try to manage the effect of software patents by specific patent clauses¹⁴⁹ or by several sentences distributed in the license text¹⁵⁰. But why does the OSLiC have to deal with this topic, if the OSLiC does not want to participate in general discussions?

In opposite to the other conditions of the open source licenses, their patent clauses or propositions in general do not directly refer to a specific set of actions which has to be executed for acting in accordance to the licenses. Open source patent clauses normally do not join in the game 'paying by doing'. So, actually, it does not seem to be not necessary to mention the patent clauses, here.

Unfortunately, although the patent clauses do not directly say '*do this or that in these or those circumstances*', some of them nevertheless trigger side effects which evoke that the distributors of open source software implicitly having already something done if they actually are distributing a piece of open source software. This implicit effect makes it necessary to deal with the patent clauses even in an only pragmatic OSLiC.

Patent clauses in open source licenses can have two different directions of impact. They use two methods to protect the users of the open source software – and sometimes these methods are combined:

- First, an open source license can assure that all contributors / distributors to / of a piece of open source software grant to all users / recipients not only the right to use the open source software itself, but automatically and implicitly also the right to use all those patents – belonging to the contributors / distributors – which as patents are necessary to use the software legally¹⁵¹. So, let us - a little simplifying and therefore only on the following few pages – name such licenses the *granting licenses*.

¹⁴⁸⁾ cf. *Jaeger a. Metzger*: Open Source Software. Rechtliche Rahmenbedingungen der Freien Software, 2011, p. 235.

¹⁴⁹⁾ pars pro toto cf. *Open Source Initiative*: APL-2.0, 2004, wp. §3.

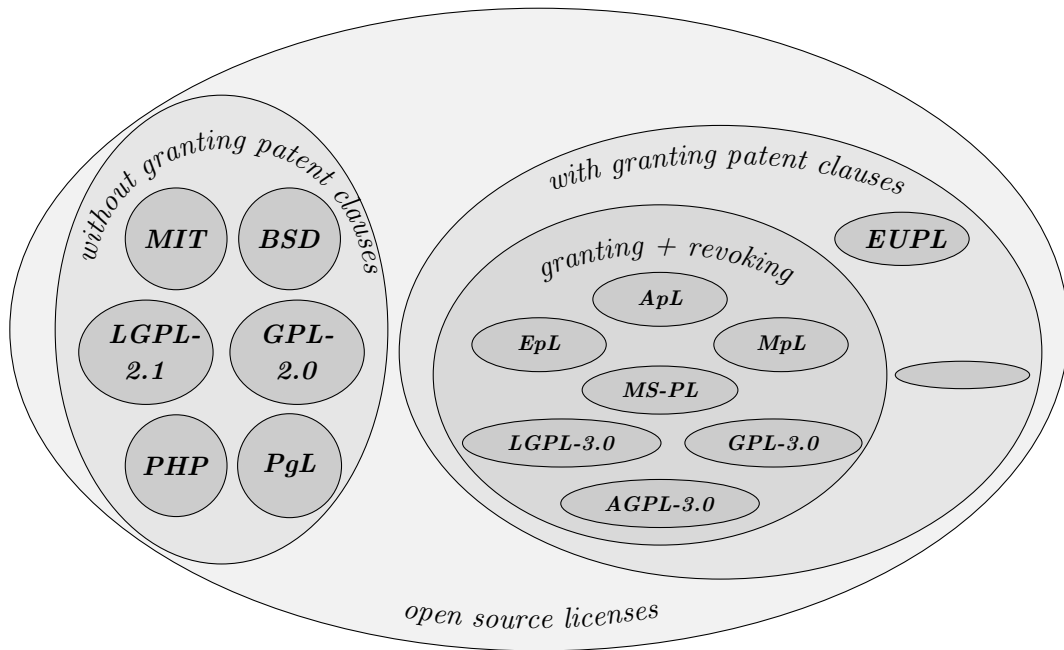
¹⁵⁰⁾ pars pro toto cf. *Open Source Initiative*: EPL-1.0, 2005, wp..

¹⁵¹⁾ There might arise a legal discussion whether a distributor who does not contribute to the software development really implicitly also has to grant the necessary rights of his patent portfolio. The OSLiC doesn't want to participate in this discussion. We take a simple and pragmatic position: for being sure that you are acting according to an open source license with such a patent clause you should simply assume that you have to do so. If this default position is not reasonable for you it might be a good idea to consult legal experts which – perhaps – may find another way for you to use the software legally.

3 Open Source: About Some Side Effects

- Second, an open source license can try to automatically terminate the right to use, to modify, and to distribute the software if its user initiates litigations against any of the contributors / distributors with respect to an infringement of patent. That can be seen as a revocation of earlier granted rights. So, let us name these license the *revoking licenses*.

Later on, we will summarize the concrete patent clauses of all the licenses discussed in the OSLiC as a proof for the following classification:



But regardless of the final textual form a license is using to express its granting or revoking positions, in any case one has to consider some aspects:

- Overall, one has to keep in mind that of course no licensor, contributor and/or distributor can release the right to use any patents he does not own – even not if he *tries* to release them by an open source patent clause¹⁵². Implicitly touched patents of third parties not having contributed to the

¹⁵²⁾ The EURL is one of the licenses which insists on this aspect: In the second half of its patent clause, the EURL underlines that “[...] no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity”. Moreover, it explicitly adds that “[...] if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient’s responsibility to acquire that license before distributing the Program” (cf. *Open Source Initiative: EURL-1.0*, 2005, wp. §2c).

3 Open Source: About Some Side Effects

development and/or participated in the distribution can never be implicitly and automatically released on the base of such an (open source) patent clause: no rights, no right to release¹⁵³. Hence: even for those open source licenses which try to protect the users, finally the user itself must nevertheless ensure that he does not violate the patents of third parties being unwillingly touched by the way the code works or the processes in which the software is used¹⁵⁴.

- In the context of a granting license, one has also to consider that contributing to and distributing of a piece of software implicitly evokes that all patents of the contributor and/or distributor are 'given free' which are necessary to use the software as whole – including the more or less deeply embedded libraries. So, if one wants to check whether some of the core patents of one's patent portfolio are afflicted by a patent clauses (and whether one therefore better should not use / distribute the corresponding piece of open source software), one should not forget to check the embedded libraries, too.
- Finally, one has to consider in the context of a granting license that its patent clause only releases the use of the patents in the meaning of 'allowed to be used for enabling the use of the distributed software'. The patent clause does not release the patents generally. Thus, the threat of (unwillingly) releasing patents by open source software is not as large as sometimes feared: the use of the patent is only granted in combination with the software. On the one hand, you may not use the open source software without having the right to use the patent because the use of the patent is inherently necessary for using the software - regardless, whether the open source software is embedded into a larger process or not. On the other hand, you are not allowed to use patents – released by the patent clause of an open source license – without exactly that open source software which has been licensed under this open source license, because the patent clause only refers to the use of just that open source software.
- Summarized, one has to consider that the granting open source licenses automatically and implicitly enforce you to grant all the rights which are necessary to use the software legally. Open source contributors and distrib-

¹⁵³) This is an important aspect which is sometimes not considered by programmers. Inside of DTAG we had a fruitful discussion evoked by Mr. Stephan Altmeyer who – as patent lawyer – patiently explained this constraint to us.

¹⁵⁴) Sometimes, this problem of willingly or unwillingly violated third party patents is seen as a weakness of open source software. But that is not true. It is a weakness of every software. Even a commercial licensor (developer) has only the right to license the use of those patents he really owns or he has 'bought' for relicensing them. Moreover, also commercial licensors can willingly or unwillingly violate patents of other persons.

3 Open Source: About Some Side Effects

utors should know that¹⁵⁵.

- With respect to the revoking licenses, one has to consider that their patent clauses contain negative conditions which may be read as interdictions. The OSLiC will integrate these conditions into specific 'prohibits'-sections of its to-do lists.
- Finally one should mention that in some cases, the form of the revocation used by the revoking license refers to the use of the software, in other cases to the use of the patents. But nevertheless, one can reason that – from the pragmatic viewpoint of a benevolent open source software user – also this second case of patent revocation implicitly terminates the right to use the software: If the use of a patent is necessary to use a piece of software legally, one is not allowed to use the software without having the right to use the patent, too; and if the use of the patent is not necessary for using the software, then the patent is not covered by the patent clause. So, in any case, this kind of patent clauses seems to terminate the right to use / to distribute and/or to modify the software. Hence, single users as well as companies or organizations should also respect such patent clauses if they want to be sure to use open source software compliantly.

The OSLiC wants to support its readers not only to act according to the licenses in general, but also according to its patent clause. Thus, we now briefly cite and summarize the meaning of particular patent clauses:

3.1.1 ApL statements concerning patents

Titled by the headline “Grant of Patent License”, the Apache License 2.0 contains a specific patent clause being comprised of two very long and condensed sentences¹⁵⁶. Outside of this patent clause, the word *patent* is only used once again – for requiring that one “[...] must retain, in the (sources) [...] all [...] patent [...] notices [...]”¹⁵⁷.

The one core message of the ApL patent clause is the statement that “[...] each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable [...] patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work [...]”¹⁵⁸.

¹⁵⁵) Again: It might be debatable whether this is also valid for the distributors which do not contribute anything to the development. That’s a legal discussion the OSLiC do not wish to participate in. From the viewpoint of an open source user who only wants to have one reliable and secure way to use open source software compliantly, one should perhaps assume that there is no difference.

¹⁵⁶) cf. *Open Source Initiative*: APL-2.0, 2004, wp. §3.

¹⁵⁷) cf. id., l.c., wp. §4.3.

¹⁵⁸) cf. id., l.c., wp. §3. “Contributor”, “Work” and “You” are defined §1: *Contributor* refers

3 Open Source: About Some Side Effects

The second core message of the ApL patent clause is the statement that “if You institute patent litigation against any entity [...] alleging that the Work [...] constitutes [...] patent infringement, then any patent licenses granted to You [...] shall terminate [...]”¹⁵⁹.

The third message of the ApL patent clause is the statement, that the “[...] license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted”¹⁶⁰.

Thus, the ApL is - as we are using to say in this chapter - a granting and a revoking license: At first you are granted to use all patents of all contributors which are necessary to use the software legally. But if you – with respect to the software – install any litigation concerning an infringement of patents, then the rights granted to you are revoked.

3.1.2 EPL statements concerning patents

The Eclipse Public License treats the patents being necessary to use the program in the same section and under the same headline “Grant of Rights” like all the other rights: First, the EPL clearly states that “[...] each Contributor [...] grants (the recipient) a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form”¹⁶¹. Then the EPL delimits the extend of this act of granting: Neither hardware patents of the contributors are covered by this releasing patent clause, nor patents that concern aspects out of the area of the initially intended software combination¹⁶². Finally, the EPL hints to the general fact that 3rd. party patents not belonging to the contributors can never be implicitly be released by such a patent clause. Moreover, it alleges the example that “[...] if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient’s responsibility to acquire that license before distributing the Program”¹⁶³.

Like other open source licenses, also the EPL announces at its end that “if (a) Recipient institutes patent litigation against any entity [...] alleging that the Program [...] infringes such Recipient’s patent(s), then such (granted) Recipi-

to the original licensor and to all others whose contributions have been incorporated into the Work. The *Work* denotes the result of the development process regardless of its form. *You* denote the licensees..

¹⁵⁹) cf. *Open Source Initiative*: APL-2.0, 2004, wp. §3.

¹⁶⁰) cf. id., ibid.

¹⁶¹) cf. *Open Source Initiative*: EPL-1.0, 2005, wp. §2.b.

¹⁶²) cf. id., ibid.

¹⁶³) cf. id., l.c., wp. §2.c.

3 Open Source: About Some Side Effects

ent's rights [...] shall terminate [...]"¹⁶⁴.

Thus, also the EPL is - as we are using to say in this chapter - a granting and a revoking license: At first you are granted to use all patents of all contributors which are necessary to use the software legally. But if you – with respect to the software – install any litigation concerning an infringement of patents, then the rights granted to you are revoked.

3.1.3 EUPL statements concerning patents

The European Union Public License contains a very brief patent clause. It 'only' states, that "the Licensor grants to the Licensee royalty-free, non exclusive usage rights to any patents held by the Licensor, to the extent necessary to make use of the rights granted on the Work under this Licence"¹⁶⁵. Furthermore the EUPL does not contain any patent specific revoking clause, but only an abstract clause requiring that all "[...] the rights granted hereunder will terminate automatically upon any breach by the Licensee of the terms of the Licence"¹⁶⁶. Thus, the EUPL is - as we are using to say in this chapter - a granting license and not a revoking license.

3.1.4 GPL statements concerning patents

Although the GPL versions 2.0 and 3.0 are aiming for the same results, they heavily differ with respect to textual and arguing structure. Therefore, it should be helpful to treat these two licenses separately.

3.1.4.1 GPL-2.0

The GPL-2.0 does not contain any specific patent clause by which it would grant (and revoke) the rights to use those patents belonging to the contributors and being necessary to use the software in accordance with the legal patent system.

Instead of this, the preamble of the GPL-2.0 alleges that "[...] any free program is threatened constantly by software patents" and that the authors of the GPL – for tackling this threat – "[...] had made it clear that any patent must be licensed for everyone's free use or not licensed at all"¹⁶⁷. Unfortunately, this specification is only an indirect claim which needs a lot of arguing for establishing a protective

¹⁶⁴) cf. id., l.c., wp. §7.

¹⁶⁵) cf. *European Community a. European commission Joinup*: EUPL-1.1/EN, 2007, wp. §2 at its tail.

¹⁶⁶) cf. id., l.c., wp. §12.

¹⁶⁷) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp., Preamble.

3 Open Source: About Some Side Effects

effect against patent disputes. Howsoever, this paragraph of the GPL-2.0 does not directly grant any rights to the software users to use necessary patents, too.

With respect to the patent problem, the GPL-2.0 also states that a licensee has to fulfill the conditions of the GPL-2.0 completely, even if an existing patent infringement – being “imposed” on the GPL licensee – “[...] contradicts the conditions of this license” so, that a waiver of the use of the software is the only way to fulfill both constraints¹⁶⁸. And finally the GPL-2.0 allows the original copyright holder to “add an explicit geographical distribution limitation excluding [...] countries” provided that these countries “[...] (re)strict the distribution and/or use of the library [...] by patents [...]”¹⁶⁹. Based on these statements, one cannot infer that the GPL-2.0 grants any patent rights to the software user, neither directly, nor indirectly.

Thus, the GPL-2.0 is not a granting or a granting and revoking license.

3.1.4.2 GPL-3.0

Initially, the GPL-3.0 regrets that “[...] every program is threatened constantly by software patents” what should be seen as the “[...] danger that patents applied to a free program could make it effectively proprietary”. And therefore – as the GPL-3.0 itself summarizes its patent rules – “[...] the GPL assures that patents cannot be used to render the program non-free”¹⁷⁰. This kind of protection is then established by three steps. First, the GPL-3.0 stipulates that “each contributor grants [...] the licensees] a 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”¹⁷¹. Second, the GPL-3.0 defines that this patent license granted by the contributor “[...] is automatically extended to all recipients” who later on receive any version of the work, even if they indirectly receive them by third parties and even if they receive a “covered work” or “works based on it”¹⁷². Moreover, the GPL-3.0 also specifies that those distributors of a “covered work” who have the right to use a patent necessary for the use of the distributed software but who are not allowed to relicense this patent to third parties must solve this problem by making the source code available nevertheless, by “depriving” themselves or by “extending the patent license to downstream recipients”¹⁷³. And

¹⁶⁸) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp. §11.

¹⁶⁹) cf. id., l.c., wp. §12.

¹⁷⁰) cf. *Open Source Initiative*: GNU General Public License, version 3 (GPL-3.0); 2007 [n.y. of the html page itself] [URL: <http://opensource.org/licenses/GPL-3.0>] – reference download: 2013-03-05, wp. Preamble.

¹⁷¹) cf. id., l.c., wp. §11.

¹⁷²) cf. id., ibid.

¹⁷³) cf. id., ibid.

3 Open Source: About Some Side Effects

finally, the GPL-3.0 also introduces an revoking clause by stating that a licensee “[...] may not initiate litigation [...] alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it”¹⁷⁴ and that this licensee “automatically” loses the rights granted by the GPL-3.0 – “including any patent licenses” – if he tries to propagate or modify a covered work against the rules of the GPL-3.0¹⁷⁵.

Thus, GPL-3.0 is - as we are using to say in this chapter - a granting and a revoking license: At first, one is granted the right to use all patents of all contributors which are necessary to use the software legally. But if one – with respect to the software – install any litigation concerning an infringement of patents, then the rights granted to you are revoked.

3.1.5 LGPL statements concerning patents

As already mentioned above, the LGPL versions 2.1 and 3.0 heavily differ with respect to textual and arguing structure. Therefore, they should be treated separately.

3.1.5.1 LGPL-2.1

Like the GPL-2.0, the LGPL-2.1 does not contain any specific patent clause by which it would grant (and revoke) the rights to use those patents belonging to the contributors and being necessary to use the software in accordance with the legal patent system.

Instead of this, the preamble of the LGPL-2.1 says that “[...] software patents pose a constant threat to the existence of any free program” and that the authors of the LGPL – for tackling this threat – “[...] insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license”¹⁷⁶. Unfortunately, this specification is again only an indirect claim which needs a lot of arguing for establishing a protective effect against patent disputes. However, this paragraph of the LGPL-2.1 does not directly grant any rights to the software users to use necessary patents too.

With respect to the patent problem, the LGPL-2.1 also states that a licensee has to fulfill the conditions of the LGPL-2.1 completely, even if an existing patent infringement – being “imposed” on the LGPL licensee – “[...] contradicts the

¹⁷⁴) cf. *id.*, l.c., wp. §10.

¹⁷⁵) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. §8.

¹⁷⁶) cf. *Open Source Initiative*: The GNU Lesser General Public License, version 2.1 (LGPL-2.1); 1999 [n.y. of the html page itself] (URL: <http://opensource.org/licenses/LGPL-2.1>) – reference download: 2013-03-06, wp., Preamble.

3 Open Source: About Some Side Effects

conditions of this license” so that a waiving of the use of the software is the only way to fulfill both constraints¹⁷⁷. And finally the LGPL-2.1 allows the original copyright holder to “add an explicit geographical distribution limitation excluding [...] countries” provided that these countries “[...] (re)strict the distribution and/or use of the library [...] by patents [...]”¹⁷⁸. Based on these statements, one cannot infer that the LGPL grants any patent rights to the software user, neither directly, nor indirectly.

Thus, also the LGPL-2.1 is not a granting or a granting and revoking license.

3.1.5.2 LGPL-3.0

The LGPL-3.0 is an extension of the GPL-3.0. Before starting with a section “Additional Definitions”, the LGPL-3.0 states that it “[...] incorporates the terms and conditions of version 3 of the GNU General Public License” and then “supplements” this GPL-3.0 content by some “additional permissions”¹⁷⁹. The LGPL-3.0 itself does not contain the word ‘patent’, but the GPL-3.0¹⁸⁰. So, the LGPL-3.0 inherits its patent clause from the GPL-3.0 which is - as we already described¹⁸¹ - a granting and a revoking license.

3.1.6 MPL statements concerning patents

The MPL distributes its statements concerning the tolerated use of the patents over three paragraphs: First, it clearly says that “each Contributor [...] grants [...] the licensee] a world-wide, royalty-free, non-exclusive license [...] under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version”¹⁸². Second, it highlights some “limitations”¹⁸³. And finally, the MPL introduces a revoking clause which signifies that the rights, granted to the licensee “[...] by any and all Contributors [...] shall terminate” if the licensee “initiates litigation against any entity by asserting a patent infringement claim [...] alleging that a Contributor Version directly or indirectly infringes any patent [...]”¹⁸⁴.

¹⁷⁷) cf. id., l.c., wp. §11.

¹⁷⁸) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp. §12.

¹⁷⁹) cf. *Open Source Initiative*: The GNU Lesser General Public License, version 3.0 (LGPL-3.0); 2007 [n.y. of the html page itself] (URL: <http://opensource.org/licenses/LGPL-3.0>) – reference download: 2013-03-06, wp..

¹⁸⁰) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. §11.

¹⁸¹) → OSLiC, p. 45

¹⁸²) cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp. §2.1, esp. §2.1.b.

¹⁸³) cf. id., l.c., wp. §2.1, esp. §2.3.

¹⁸⁴) cf. id., l.c., wp. §5.2.

Thus, the MPL is - as we are using to say in this chapter - a granting license and a revoking license.

3.1.7 MS-PL statements concerning patents

First, the MS-PL contains a statement, by which “[...] each contributor grants (the software users) a non-exclusive, worldwide, royalty-free license under its licensed patents to make, have made, use, sell, offer for sale, import, and/or otherwise dispose of its contribution in the software or derivative works of the contribution in the software”¹⁸⁵. Second, the MS-PL says that “if you bring a patent claim against any contributor[...] your patent license from such contributor to the software ends automatically”¹⁸⁶.

Thus, the MS-PL is - as we are using to say in this chapter - a granting and a revoking license: At first you are granted to use all patents of all contributors which are necessary to use the software legally. But if you – with respect to the software – install any litigation concerning an infringement of patents, then the rights granted to you are revoked.

3.2 Excursion: What is a 'Derivative Work' - the basic idea of open source

This chapter briefly discusses existing attempts to define the derivated works of technical aspects, like dynamical or statical linking or not. We will prove that linking can not deliver a definite criteria: 1) modules are only unzipped libraries. 2) you can distribute software as modules added by a script, which statically(sic!) links all modules before executing the program. 3) The criteria of pipe-communication is good, but not sufficient. 4) All these attempts do not match the constituting features of script languages. Therefore we will follow Moglen(?) and will argue from the viewpoint of a developer: it is only a question of a function, method or anything else which calls (jumps into) a piece of code which has been licensed by a license protecting on-top-developments and you have a derivated work.

...

As first version, we present an outline of the arguing structure this chapter will later present in a more narrative way.

The meaning 'derivative work' must be known! Many open source licenses are using the the term 'derivative work' [cite the sources], either direct or indirect in form of the work 'modification' [Write a table as survey]. And nearly

¹⁸⁵) cf. *Open Source Initiative*: MS-PL, 2013, wp. §2.B.

¹⁸⁶) cf. id., l.c., wp. §3.B.

3 Open Source: About Some Side Effects

all licenses, which are using the term 'derivative work' etc., are linking tasks which must be executed to comply with the corresponding license, to the precondition, that something is a derivative work. [tabke survey]. **Hence, for acting in accordance to such a license, it has to be known, what a derivate work is**

Unfortunately the meaning is not clearly fixed . The exist some different readings of the term 'derivative work' [specify the differences and cite the sources] **Hence, it is not as clear as wished what a derivtive work is**

Hence lets argue from the viewpoint of a benevolent developer : Open source licenses are written for software developers, mostly to preserve their freedom, to develop software. And sometimes these licenses are also written by software developers – or at least by the assistance of. So, one should be able to answer the question under which circumstances a piece of software is a 'deverivate work' of another piece of software on the base of two principles:

- Let us argue on the base of a benevolent neutral software developer without hidden interests or a hidden agenda.
- In case of doubts let us preferably assume that the two pieces interrelate as source and derivative work – so that the OSLiC rather recommends to execute the required tasks than to put them away.

This are our rules by which the OSLiC decides to take something as derivative Work:

Copy-Case Copying a piece of code from a source file and pasting it into a target file makes the target file a derivate work of the source file¹⁸⁷.

Modify-Case Inserting any new content or deleting any existing content of a source file makes the resulting target file being a derivate work of the wource file.

Call-Case Inserting into a target file the call of function which is defined inside of and delivered by a sourcefile makes the target file depending on the source file and therefore a derivative work of the delivering source file.

And here are some applications of the rules

- Combining the copy-case and the modify-case on the base of two different source files make the resulting target file being a derivative work of the two source files (follows from copy-case and modify-case)
- ...

¹⁸⁷⁾ Be careful: this case must still be distinguished from the case of an automatically inclusion (header files, script libraries) during the compilation / execution: Header files allon should not evekoe a derivative work.

3.3 Excursion: The problem of license compatibility [tbd]

Here we discuss the often neglected or only loosely touched problem of combining differently licensed software. We will hint to the Exclusion-List of the Free software foundation; we will hint to the Eclipse / GPL-plugin problem; we will mention the recent discussion whether the kernel requires to license the complete Android as GPL; and finally we will discuss the just now published, short analysis of Jaeger and Metzger presenting a combining matrix which seems to fall into their lap. We ourselves will argue that the question can simply be answered: only if you embed two libraries which both are licensed by an on-top-development protecting license and if these both licenses require the licensing of the derivated work by different licenses then you have a problem. In all other cases which we will describe there is no problem.

...

3.4 Excursion: open source software and money [tbd]

Here we will shortly discuss ways in which money and Open Source is no problem.

...

4 Open Source Use Cases: Concept and Taxonomy

This chapter establishes our concept of open source use cases as a classification system for to-do lists. The conditions of a specific license, in the context of a particular open source use case, shall be satisfiable by following the corresponding to-do list. Additionally this chapter introduces a taxonomy for these open source use cases. Later on, this taxonomy will organize the Open Source Use Case Finder.

After all these introductory remarks, we can summarize our idea. We know that the right to use open source software depends on the tasks required by the open source licenses. As opposed to commercial licenses, you can not buy the right to use a piece of open source software by paying money. It is embedded into the *Open Source Definition* that the right to use the software may not be sold. The OSD states firstly that an open source license may “[...] not restrict any party from selling or giving away the software as a component of (any) aggregate software distribution”, and adds secondly in the same context that an open source license “[...] shall not require a royalty or other fee for such sale”¹⁸⁸.

However, it would be wrong to conclude that you are automatically allowed to use open source software without any service in return: generally you have to do something to gain the right to use the software. In other words: open source software is covered by the idea of ‘paying by doing’. Accordingly, open source licenses describe specific circumstances under which the user must execute some tasks in order to be compliant with the licenses. So, if we want to offer to-do lists for fulfilling license conditions, we must consider these tasks and circumstances.

In practice, such circumstances are not linear and simple. They contain combinations of (sometimes context sensitive) conditions which can be grouped into classes of tokens. Such a class of tokens might denote a feature of the software itself – such as being an application or a library. Or it can refer to the circumstances of using the software, such as ‘using the software only for yourself’ or ‘distributing the software also to third parties’.

At the end, we want to determine a set of specific OSUCs – the *open source use cases*. And we want to deliver for each of these OSUCs and for each of the considered open source licenses one list of actions which fulfills the license in that context¹⁸⁹.

Such an *open source use case* shall be considered as a set of tokens describing the

¹⁸⁸⁾ cf. *Open Source Initiative*: The Open Source Definition, 2012, wp. §1.

¹⁸⁹⁾ Fortunately, sometimes one task list fulfills the conditions of more than one use case – a welcome reduction of complexity

4 Open Source Use Cases: Concept and Taxonomy

circumstances of a specific usage. Hence, to begin, we must specify the relevant classes of tokens, before we can determine the valid combinations of these tokens – our *open source use cases*. Finally, based on the tokens, we generate a taxonomy in the form of a tree. This tree will become the base of the *Open Source Use Case Finder* which will be offered in the next chapter, and which leads you to your specific OSUC by evaluating just a few questions and answers.

There are only a handful of tokens which are relevant to the circumstances of open source software licenses:

- The **type of the open source software**: On the one hand, we regard code snippets, modules, libraries and plugins, and on the other hand, autonomous applications, programs and servers. We will take the word 'snimolis' for the first set, and 'proapses' for the second. This is necessary, as we are not only talking about libraries and applications in the everyday sense, but rather in the broadest sense¹⁹⁰. More specifically, we will ask you, whether the open source software you want to use, is an includable code snippet, a linkable module or library, or a loadable plugin, or whether it is an autonomous application or server which can be executed or processed. In the first case, the answer should be 'it is a snimoli', in the second 'it is a proapse'.
- The **state of the open source software**: It might be used exactly as one has received it. Or it can be modified, before being used. More specifically, we will ask you, whether you want to leave the open source software as you have received it, or whether you want to modify it before using and/or distributing it to 3rd parties. In the first case, the answer should be 'unmodified', in the second 'modified'.
- The **usage context of the open source software**: On the one hand you might use the received open source software as a readily prepared application. On the other hand you might embed the received open source into a larger application as one of its components. More specifically, we will ask you, whether you are using the open source software as an autonomous piece of software, or whether you are using it as an embedded part of a larger, more complex piece of software. In the first case, the answer should be 'independent', in the second 'embedded'.
- The **recipient of the open source software**: Sometimes you might wish to use the received open source software only for yourself. In other cases

¹⁹⁰⁾ Of course, our newly introduced concepts of 'snimoli' and 'proapse' are not absolutely one of the most elegant words. So, initially we tried to talk about 'applications' and 'libraries', although in our context these words should denote more, than they traditionally do. But we couldn't minimize the irritations of our interlocutors. Too often we had to remind them that we were not talking about applications and libraries in the strict sense of the words. Finally we decided to find our own words – and to stay open for better proposals ;-)

4 Open Source Use Cases: Concept and Taxonomy

you might intend to hand over the software (also) to other people. More specifically, we will ask you, whether you are going to use the open source software only for yourself, or whether you plan to (re)distribute it (also) to third parties. In the first case, the answer should be '4yourself', in the second '4others'.

- The **mode of combination**: In this case, we will ask you, whether you are going to combine or to embed the open source software with other software components by linking them statically or dynamically, or by textually including (parts of) the open source software into your larger product. In the first case, the answer should be 'statically linked', in the second 'dynamically linked', in the third 'textually included'

From a more programmatic point-of-view, we can summarize these tokens as follows:

- `type::snimoli` *or* `type::proapse`
- `state::unmodified` *or* `state::modified`
- `context::independent` *or* `context::embedded`
- `recipient::4yourself` *or* `recipient::4others`
- `mode::statically-linked` *or* `mode::dynamically-linked` *or* `mode::textually-included`

We already defined an open source use case as a combination of these tokens. If we simply combine all these tokens of all these classes with all the tokens of the other classes¹⁹¹, we get $2*2*2*2*3 = 48$ sets of tokens – or 48 *open source use cases*. Fortunately, some of the generated sets are invalid from an empirical or logical view, and some of these sets are context sensitive:

1. It would be unreasonable to ask you whether you are going to combine the received software with other software components by linking them statically or dynamically, or by including it textually into a larger unit, if you already have answered that the received open source software is a *proapse* or that it shall be used *independently*: A readily prepared application or server can't be linked to another application or server which also contains a **main**-function. And using a *proapse* or *snimoli* *independently* implies that it is not used *in combination* with other units.
2. If you already have specified that the used open source software is a *proapse* – an autonomous program, an application, or a server – then your answer implies that the software is used independently and is not embedded with other components into a larger unit. But if you have specified that the used

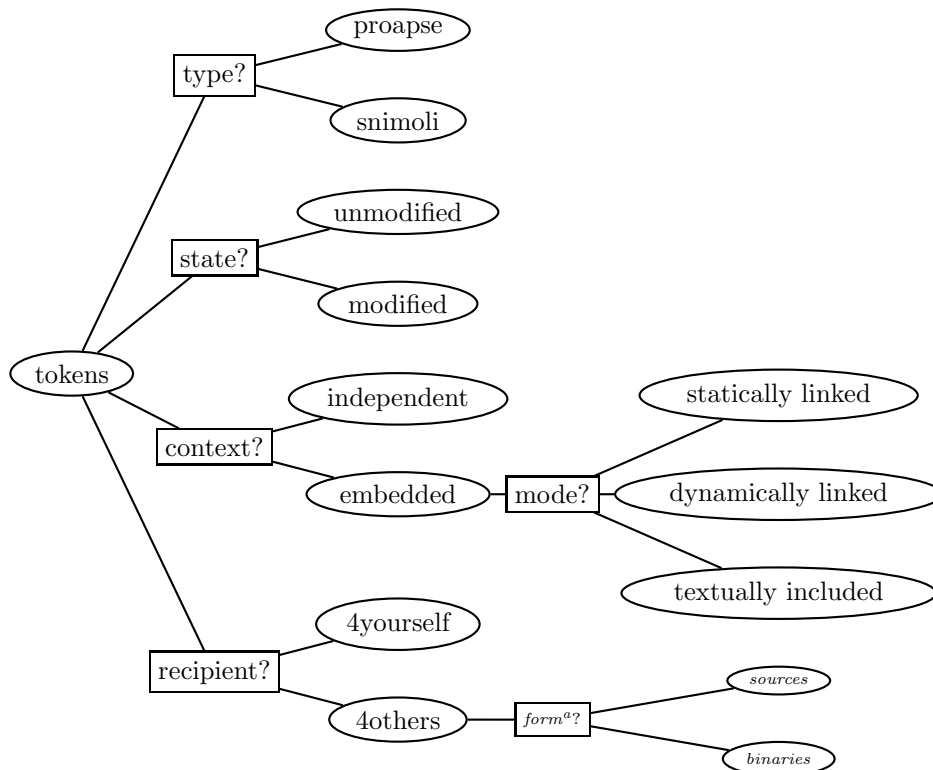
¹⁹¹) in the sense of the cross product $\text{TYPE} \times \text{STATE} \times \text{CONTEXT} \times \text{RECIPIENT} \times \text{MODE}$

open source software is a *snimoli* – a snippet of code, a module, a plugin, or a library – then it can indeed be used as an embedded component of a constructed larger application or server, or it can be used independently in case you 'only' re-distribute it to 3rd. parties.

3. If you already have specified that the used open source software is a *snimoli* – a snippet of code, a module, a plugin, or a library – and that this *snimoli* shall be used only by yourself (not distributed to other 3rd. parties) then your answer must also imply that this *snimoli* is used in combination, as an embedded part of a larger unit. A library can not be used autonomously, without using it as a component of another application. In this case, it would simply sit on the disk and would do nothing more than occupying space.

Does this sound complex? We thought so, too. We spent much time explaining these constraints to ourselves, and only when we had transposed all the combinations and rules into a tree, the situation became clearer. The following diagrams shall summarize this way of clarification:

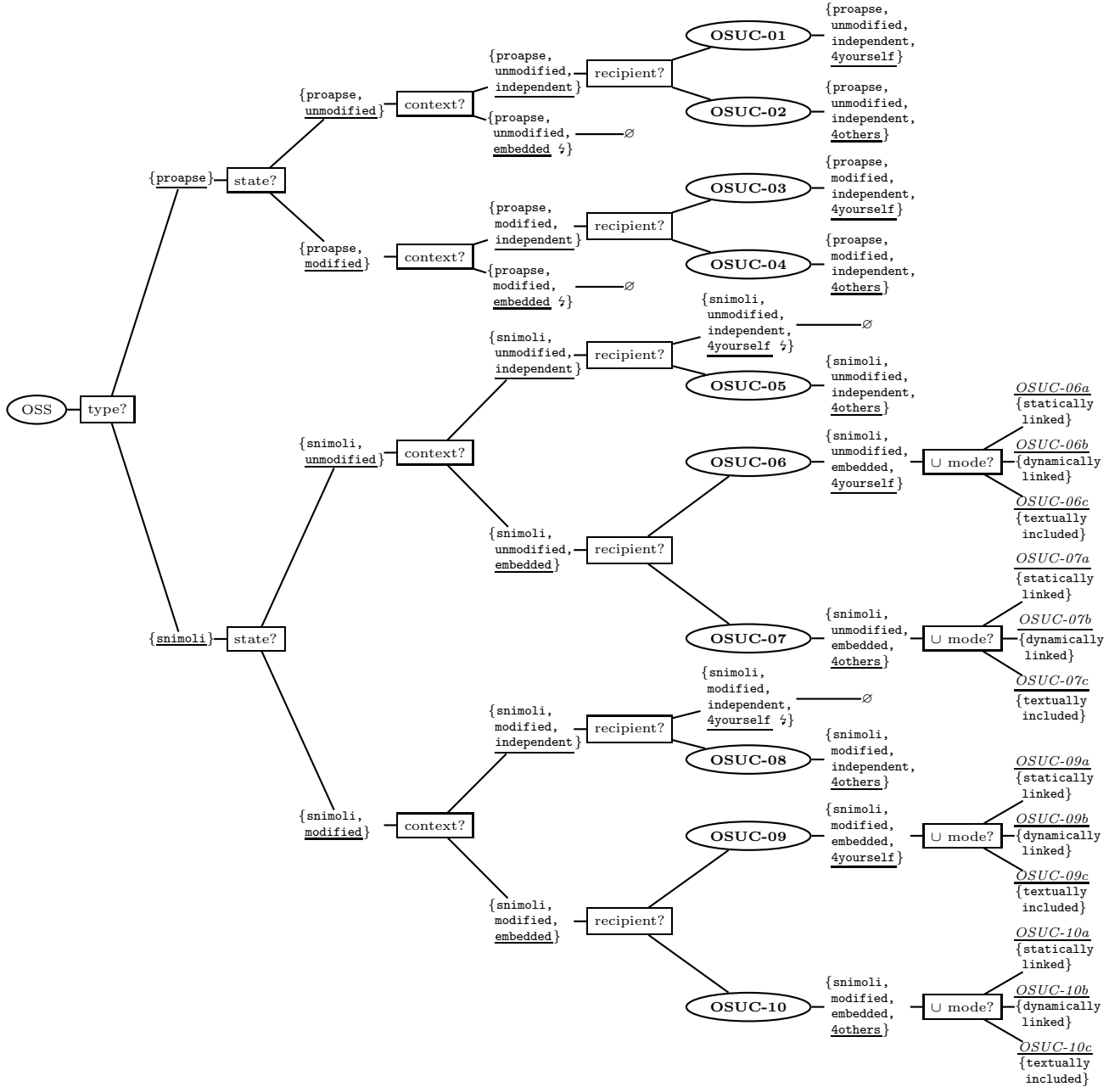
4.1 Overview of the OSUC classes and tokens



^{a)} For differentiating between distributing sources and binaries → OSLiC, p. 56

4.2 The OSUC taxonomy

This is one of the possible trees 'collecting' the tokens and offering the *open source use cases* as their leafs¹⁹²:



¹⁹²) Each of the invalid use cases (= sets of tokens) [for details s. p. 53] is marked by an ⚡ and leads to an empty set (= ∅): A proapse can not be embedded with another software unit, also containing a main-function. Using a software library only for yourself and independent (not in combination with larger software unit), is like having an unused heap of bytes on your disc.

4.3 About a suppressed degree of complexity

Many licenses also draw a distinction between distributing the software as sources and distributing them as binaries. This aspect is not covered by the OSLiC taxonomy. Thus, for being truly complete, all the branches of the taxonomy referring to the token *4others* should have been split by tokens like *distributing sources* and *distributing binaries*. But this would have made the taxonomy unusable. So, we suppressed these criteria. But – of course – we do not want to reduce the reality inadequately; we want to solve the problem in a suitable manner. Therefore, we have implicitly split all relevant use cases by writing two to-do lists for them, one concerning the *distribution of sources* and the other the *distribution of binaries*. Please select the relevant to-do list based on its title.

5 Open Source Use Cases: Find the License Fulfilling To-do Lists

This chapter offers the Open Source Use Case Finder: First, it presents a form to gather the specifying information. Second, it offers a tree which can easily be traversed using the gathered information. And finally it contains the list of open source use cases. Each leaf of the tree leads to one open source use case which itself then refers to the specific license fulfilling to-do lists.

5.1 A standard form for gathering the relevant information

	<i>Which open source software do you want to use?</i>	
	<i>Under which open source license is it released?</i>	
Class	Questions	Answers
Type	<i>Is the open source software you want to use a software library in the broadest sense (an includable code snippet, a linkable module or library, or a loadable plugin) [=snimoli], or is it an autonomous program, application or server which can be executed or processed [=proapse]?</i>	<input type="checkbox"/> snimoli <input type="checkbox"/> proapse
State	<i>Do you want to leave your open source software as you have got it, or do you want to modify it before using and/or distributing it to 3rd parties?</i>	<input type="checkbox"/> unmodified <input type="checkbox"/> modified
Context	<i>Are you using your open source software as an autonomous piece of software [=independent], or are you using it as an embedded part or component of a larger, more complex piece of software [=embedded]?</i>	<input type="checkbox"/> independent <input type="checkbox"/> embedded
Recipient	<i>Are you going to use the received open source software only for yourself [=4yourself], or do you plan to (re)distribute it (also) to third parties [=4others]?</i>	<input type="checkbox"/> 4yourself <input type="checkbox"/> 4others
Form	<i>If you plan to (re)distribute an open source based work [=4others], do you want to distribute only the binaries or (also) the source code?</i>	<input type="checkbox"/> only binaries <input type="checkbox"/> (also) sources
Mode	<i>Are you going to combine the received open source software with other software components by linking all together statically, by linking them dynamically, or by textually including (parts of) the open source software into your larger unit?</i>	<input type="checkbox"/> statically linked <input type="checkbox"/> dynamically linked <input type="checkbox"/> textually included

As discussed earlier, there are of course some invalid combinations¹⁹³. Here are some extra explanations about each class:

Type: A piece of (open source) software shall be viewed as a program, an application, or a server, if you can start its binary form with your normal program launcher, or (in case of a text file which still must be interpreted by an interpreter like php, perl, bash etc.) if you can start an interpreter taking the file as one of its arguments.

State: You modify open source software if you expand, reduce or modify at

¹⁹³⁾ type::proapse excludes state::embedded; recipient::4yourself excludes the combination with state::independent and type::snimoli; any value of class 'mode' implies state::embedded [for details see page 53]. If you have gathered one of these invalid combinations, please check the corresponding explanations

5 Open Source Use Cases: Find the License Fulfilling To-do Lists

least one of the received software files, and – in case of dealing with binary object code – if you (re)compile and (re)link the modified software to a new binary file. If you only modify configuration files, you do not modify the open source software.

Context: You use open source software embedded into a larger unit, if one of your files of the larger unit contains a verbatim or modified copy (i.e. a snippet) of the received open source software, or if the larger unit contains an include statement referring to a file of the received open source software, or if your development environment contains a compiler or linker directive referring to the received open source software.

Recipient: You use the received open source software only for yourself, if you as a person do not pass it to other entities like persons, organizations, companies etc., or if you – as a member of a specific development group – pass it only to the other members of your development group. But if you store open source software on any device such as a mobile phone, an USB stick, etc. or if you attach it to any transport medium like email etc. and if you then sell, give away, or simply send this device or transport medium to anyone (other than a direct member of your development group) then you indeed handover the open source software to third parties¹⁹⁴.

Form: Often it is up to you to decide whether you want to distribute the source code, too. But if you do so, you have to respect special conditions¹⁹⁵.

Mode: The definition follows the corresponding standard terms of the software development.

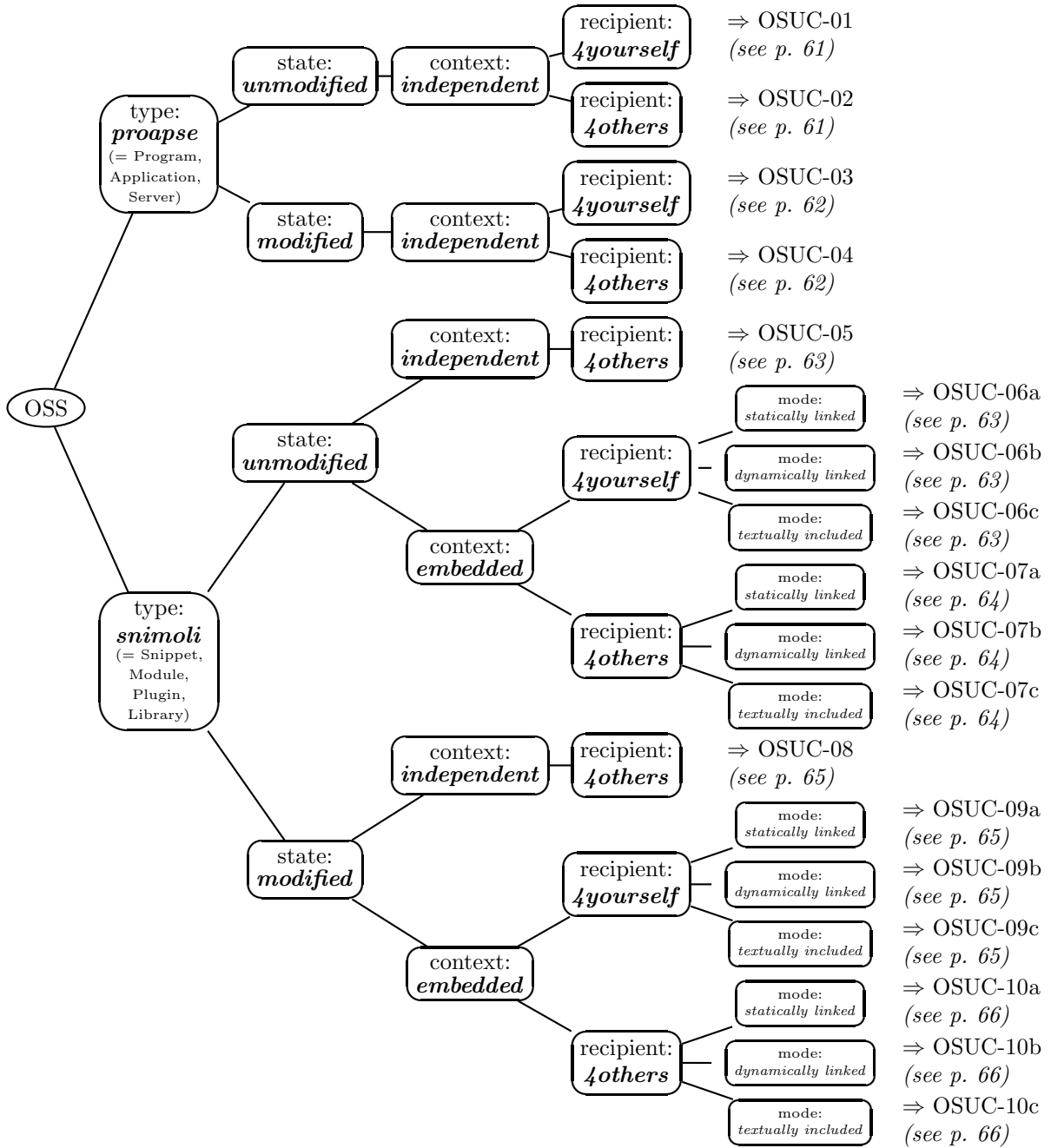
5.2 The taxonomic Open Source Use Case Finder

Now, after having gathered the necessary information, determine your specific open source use case by traversing the following tree and its corresponding branches:

¹⁹⁴⁾ Please remember that – at least in Germany – there are opinions that even handing over software to another legal entity or department of the same company is also a kind of distribution. It is always safest to take the broadest possible meaning of distributing or handing over.

¹⁹⁵⁾ For details concerning a necessary refinement of the open source use case taxonomy, please see → OSLiC, p. 56

5 Open Source Use Cases: Find the License Fulfilling To-do Lists



Please keep in mind, that for reducing the complexity of this general taxonomy, we suppressed the differentiation between distributing binaries and distributing sources¹⁹⁶. Nevertheless, the final to-do lists respect this difference wherever it is necessary. It is up to you to select the correct to-do list on the basis of its titles.

¹⁹⁶) For details concerning a necessary refinement of the taxonomy → OSLiC, p. 56

5.3 The open source use cases and its to-do list references

On the following pages, each **Open Source Use Case** is textually specified one more time and added by a list of page numbers. Each of these pages hints to that license-specific to-do list whose items together offer a processable way for acting according to the license under the circumstances of the described **Open Source Use Case**.

OSUC-01: Only for yourself, you are using an unmodified open source program, application, or server – just as you received it. You are not going to combine it with other components in the sense of software development (= *proapse, unmodified, independent, 4yourself*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 100 for the **AGPL** (= *Affero GNU Public License*)
- p. 68 for the **ApL** (= *Apache License*)
- p. 77 for the **BSD** License (= *Berkeley Software Distribution*)
- p. 98 for the **EPL** (= *Eclipse Public License*)
- p. 99 for the **EUPL** (= *European Union Public License*)
- p. 101 for the **GPL** (= *GNU Public License*)
- p. 100 for the **LGPL** (= *Lesser GNU Public License*)
- p. 86 for the **MIT** License (= *Massachusetts Institute of Technology*)
- p. 99 for **MPL** (= *Mozilla Public License*)
- p. 90 for the **MS-PL** (= *Microsoft Public License*)
- p. 96 for the **PGL** (= *Postgres License*)
- p. 97 for the **PHP** license

OSUC-02: Just as you received it, you are going to distribute an unmodified open source program, application, or server to 3rd parties. In this act of distribution, you do not combine this program, application, or server with other software components in the sense of software development (= *proapse, unmodified, independent, 4others*)¹⁹⁷. To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 100 for the **AGPL** (= *Affero GNU Public License*)
- p. 69 for the **ApL** (= *Apache License*)
- p. 78 for the **BSD** License (= *Berkeley Software Distribution*)

¹⁹⁷) For differentiating between distributing sources and binaries → OSLiC, p. 56

5 Open Source Use Cases: Find the License Fulfilling To-do Lists

- p. 98 for the **EPL** (= *Eclipse Public License*)
- p. 99 for the **EUPL** (= *European Union Public License*)
- p. 101 for the **GPL** (= *GNU Public License*)
- p. 100 for the **LGPL** (= *Lesser GNU Public License*)
- p. 86 for the **MIT** License (= *Massachusetts Institute of Technology*)
- p. 99 for **MPL** (= *Mozilla Public License*)
- p. 91 for the **MS-PL** (= *Microsoft Public License*)
- p. 96 for the **PGL** (= *Postgres License*)
- p. 97 for the **PHP** license

OSUC-03: Only for yourself, you are modifying a received open source program, application, or server, before you are using it. But you do not combine it with other components in the sense of software development (= *proapse, modified, independent, 4yourself*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 100 for the **AGPL** (= *Affero GNU Public License*)
- p. 68 for the **ApL** (= *Apache License*)
- p. 77 for the **BSD** License (= *Berkeley Software Distribution*)
- p. 98 for the **EPL** (= *Eclipse Public License*)
- p. 99 for the **EUPL** (= *European Union Public License*)
- p. 101 for the **GPL** (= *GNU Public License*)
- p. 100 for the **LGPL** (= *Lesser GNU Public License*)
- p. 86 for the **MIT** License (= *Massachusetts Institute of Technology*)
- p. 99 for **MPL** (= *Mozilla Public License*)
- p. 90 for the **MS-PL** (= *Microsoft Public License*)
- p. 96 for the **PGL** (= *Postgres License*)
- p. 97 for the **PHP** license

OSUC-04: You are going to modify a received open source program, application, or server, before you distribute it to 3rd parties. But you do not combine this modified program, application, or server with other software components in the sense of software development (= *proapse, modified, independent, 4others*)¹⁹⁸. To see the *specific, license fulfilling to-do lists* jump to the following pages:

¹⁹⁸⁾ For differentiating between distributing sources and binaries → OSLiC, p. 56

5 Open Source Use Cases: Find the License Fulfilling To-do Lists

- p. 100 for the **AGPL** (= *Affero GNU Public License*)
- p. 70 for the **ApL** (= *Apache License*)
- p. 79 for the **BSD** License (= *Berkeley Software Distribution*)
- p. 98 for the **EPL** (= *Eclipse Public License*)
- p. 99 for the **EUPL** (= *European Union Public License*)
- p. 101 for the **GPL** (= *GNU Public License*)
- p. 100 for the **LGPL** (= *Lesser GNU Public License*)
- p. 87 for the **MIT** License (= *Massachusetts Institute of Technology*)
- p. 99 for **MPL** (= *Mozilla Public License*)
- p. 91 for the **MS-PL** (= *Microsoft Public License*)
- p. 96 for the **PGL** (= *Postgres License*)
- p. 97 for the **PHP** license

OSUC-05: Just as you received it, you are going to distribute an unmodified open source library, code snippet, module, or plugin to 3rd parties. In this act of distribution, you do not combine this library, code snippet, module, or plugin with other software components in the sense of software development (= *snimoli, unmodified, independent, 4others*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 100 for the **AGPL** (= *Affero GNU Public License*)
- p. 69 for the **ApL** (= *Apache License*)
- p. 78 for the **BSD** License (= *Berkeley Software Distribution*)
- p. 98 for the **EPL** (= *Eclipse Public License*)
- p. 99 for the **EUPL** (= *European Union Public License*)
- p. 101 for the **GPL** (= *GNU Public License*)
- p. 100 for the **LGPL** (= *Lesser GNU Public License*)
- p. 86 for the **MIT** License (= *Massachusetts Institute of Technology*)
- p. 99 for **MPL** (= *Mozilla Public License*)
- p. 91 for the **MS-PL** (= *Microsoft Public License*)
- p. 96 for the **PGL** (= *Postgres License*)
- p. 97 for the **PHP** license

OSUC-06: Only for yourself and just as you received it, you are going to combine an unmodified open source library, code snippet, module, or plugin into a

5 Open Source Use Cases: Find the License Fulfilling To-do Lists

larger software unit as one of its parts. (= *snimoli, unmodified, embedded, 4yourself*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 100 for the **AGPL** (= *Affero GNU Public License*)
- p. 68 for the **ApL** (= *Apache License*)
- p. 77 for the **BSD** License (= *Berkeley Software Distribution*)
- p. 98 for the **EPL** (= *Eclipse Public License*)
- p. 99 for the **EUPL** (= *European Union Public License*)
- p. 101 for the **GPL** (= *GNU Public License*)
- p. 100 for the **LGPL** (= *Lesser GNU Public License*)
- p. 86 for the **MIT** License (= *Massachusetts Institute of Technology*)
- p. 99 for **MPL** (= *Mozilla Public License*)
- p. 90 for the **MS-PL** (= *Microsoft Public License*)
- p. 96 for the **PGL** (= *Postgres License*)
- p. 97 for the **PHP** license

OSUC-07: Just as you received it and before you will distribute it to 3rd parties together with the larger software unit, you combine an unmodified open source library, code snippet, module, or plugin into a larger software unit in the sense of software development (= *snimoli, unmodified, embedded, 4others*)¹⁹⁹. To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 100 for the **AGPL** (= *Affero GNU Public License*)
- p. 69 for the **ApL** (= *Apache License*)
- p. 78 for the **BSD** License (= *Berkeley Software Distribution*)
- p. 98 for the **EPL** (= *Eclipse Public License*)
- p. 99 for the **EUPL** (= *European Union Public License*)
- p. 101 for the **GPL** (= *GNU Public License*)
- p. 100 for the **LGPL** (= *Lesser GNU Public License*)
- p. 86 for the **MIT** License (= *Massachusetts Institute of Technology*)
- p. 99 for **MPL** (= *Mozilla Public License*)
- p. 91 for the **MS-PL** (= *Microsoft Public License*)

¹⁹⁹) For differentiating between distributing sources and binaries → OSLiC, p. 56

5 Open Source Use Cases: Find the License Fulfilling To-do Lists

- p. 96 for the **PGL** (= *Postgres License*)
- p. 97 for the **PHP** license

OSUC-08: Before you will distribute it, you are going to modify an open source library, code snippet, module, or plugin to 3rd parties, but you do not combine it with other software components in the sense of software development (= *snimoli, modified, independent, 4others*)²⁰⁰. To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 100 for the **AGPL** (= *Affero GNU Public License*)
- p. 72 for the **ApL** (= *Apache License*)
- p. 80 for the **BSD** License (= *Berkeley Software Distribution*)
- p. 98 for the **EPL** (= *Eclipse Public License*)
- p. 99 for the **EUPL** (= *European Union Public License*)
- p. 101 for the **GPL** (= *GNU Public License*)
- p. 100 for the **LGPL** (= *Lesser GNU Public License*)
- p. 88 for the **MIT** License (= *Massachusetts Institute of Technology*)
- p. 99 for **MPL** (= *Mozilla Public License*)
- p. 93 for the **MS-PL** (= *Microsoft Public License*)
- p. 96 for the **PGL** (= *Postgres License*)
- p. 97 for the **PHP** license

OSUC-09: Only for yourself, you are going to modify an open source library, code snippet, module, or plugin, before you will combine it – in the sense of software development – into a larger software unit as one of its parts. (= *snimoli, modified, embedded, 4yourself*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 100 for the **AGPL** (= *Affero GNU Public License*)
- p. 68 for the **ApL** (= *Apache License*)
- p. 77 for the **BSD** License (= *Berkeley Software Distribution*)
- p. 98 for the **EPL** (= *Eclipse Public License*)
- p. 99 for the **EUPL** (= *European Union Public License*)
- p. 101 for the **GPL** (= *GNU Public License*)
- p. 100 for the **LGPL** (= *Lesser GNU Public License*)
- p. 86 for the **MIT** License (= *Massachusetts Institute of Technology*)

²⁰⁰⁾ For differentiating between distributing sources and binaries → OSLiC, p. 56

5 Open Source Use Cases: Find the License Fulfilling To-do Lists

- p. 99 for **MPL** (= *Mozilla Public License*)
- p. 90 for the **MS-PL** (= *Microsoft Public License*)
- p. 96 for the **PGL** (= *Postgres License*)
- p. 97 for the **PHP** license

OSUC-10: Before you will distribute it to 3rd parties, you are going to modify an open source library, code snippet, module, or plugin, which you then combine with other software components in the sense of software development (= *snimoli, modified, independent, 4others*)²⁰¹. To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 100 for the **AGPL** (= *Affero GNU Public License*)
- p. 74 for the **ApL** (= *Apache License*)
- p. 82 for the **BSD** License (= *Berkeley Software Distribution*)
- p. 98 for the **EPL** (= *Eclipse Public License*)
- p. 99 for the **EUPL** (= *European Union Public License*)
- p. 101 for the **GPL** (= *GNU Public License*)
- p. 100 for the **LGPL** (= *Lesser GNU Public License*)
- p. 88 for the **MIT** License (= *Massachusetts Institute of Technology*)
- p. 99 for **MPL** (= *Mozilla Public License*)
- p. 95 for the **MS-PL** (= *Microsoft Public License*)
- p. 96 for the **PGL** (= *Postgres License*)
- p. 97 for the **PHP** license

²⁰¹⁾ For differentiating between distributing sources and binaries → OSLiC, p. 56

6 Open Source License Compliance: To-Do Lists

With respect to the defined open source use cases, this chapter lists what one has to do for acting in accordance with the specific open source licenses

6.1 Some general remarks on 'giving' someone a file

This chapter has to be started with some general hints being relevant for many to-do lists. For not repeating these remarks too often, we are starting with these general remarks and will later on refer to these remarks:

- On the one hand, sometimes you will distribute binary packages containing open source software components or complete open source applications. Moreover, in some cases, you probably want to distribute them on a medium which doesn't allow the user, to see the package files directly – some mobile devices don't give their users the full access to all stored files. On the other hand, open source licenses often require 'to give' someone copies of text files, like the license itself, copyright notes, specific notice files or anything else. The safe interpretation of 'giving someone a text' means that the receiver must be able to read it²⁰². Hence, on systems which offer a file browser and a suitable reader, it is sufficient, to put these file onto the files system. On the other systems, you *must* present the content of the files by your application – for example in a specific copyright dialog²⁰³. The OSLiC does not want to refine the taxonomies down to the level of operating systems. So, it is up to the reader to transfer these conditions into the to-do lists.
- Sometimes a product which uses and distributes open source software tries to fulfill the requirement 'to give the recipients the license etc.' by presenting links to general versions of these licensing files hosted somewhere on the internet. But be aware: Although it is a good tradition – especially if you link to the homepages of the projects for being totally transparent – it is not sufficient to offer only the links. If you are required by the open source licenses to handover something to your users, *you* must do it. It is not safe to delegate the task to anyone hoping that he will offer the files all the time

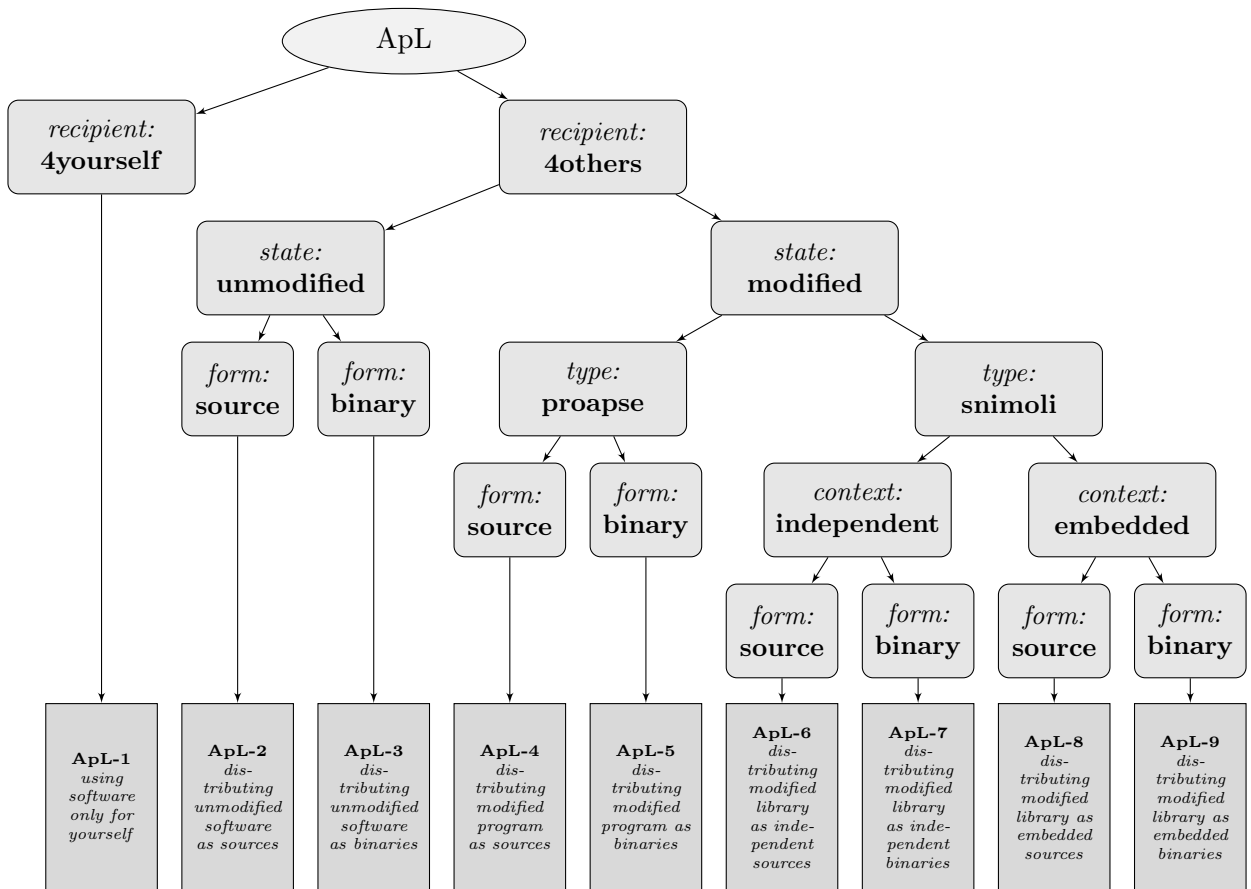
²⁰²) To give someone anything he can't touch, feel, see etc., is like not giving him the object ;-)

²⁰³) Additionally, in the open source community, it is a good tradition, to present these reference data voluntarily.

your product is distributed²⁰⁴. But even it would be save, the point is: you have to fulfill the license, no one else.

6.2 Apache licensed software

Officially, only the Apache License, Version 2.0 is an approved open source license²⁰⁵. Explicitly, it 'only' focuses on the "redistribution"²⁰⁶. So, you can also use the following simplified Apache specific open source use case finder²⁰⁷:



6.2.1 ApL-1: Using the software only for yourself

means that you are going to use a received Apache licensed software only for yourself and that you do not handover it to any 3rd party in any sense.

²⁰⁴) Moreover, the advantage of doing the job oneself is that one has not to struggle with uncommunicated implicate modifications of the link targets.

²⁰⁵) For details → OSLiC, pp. 23

²⁰⁶) cf. *Open Source Initiative: The Open Source Licenses*, alphabetically sorted, 2012, wp. §4.

²⁰⁷) For details of the general OSUC finder → OSLiC, pp. 52 and 55

covers OSUC-01, OSUC-03, OSUC-06, and OSUC-09²⁰⁸

requires no tasks in order to fulfill the conditions of the Apache 2.0 license with respect to this use case:

- You are allowed to use any kind of Apache software in any sense and in any context without any obligations premised you do not handover the software to 3rd parties.

prohibits nothing explicitly.

6.2.2 ApL-2: Passing the unmodified software as source code

means that you are going to distribute an unmodified version of the received Apache software to 3rd parties in form of a set of source code files or an integrated source code package²⁰⁹

covers OSUC-02, OSUC-05, OSUC-07²¹⁰

requires the following tasks in order to fulfill the license conditions

- **[mandatory:]** Give the recipient a copy of the Apache 2.0 license. If it is still not incorporated into the software package, add it²¹¹.
- **[mandatory:]** Ensure that the licensing elements – esp. the specific copyright notice of the original author(s) – are retained in your package in the form you have received them.
- **[mandatory:]** Ensure that a *notice text file*²¹² is retained in your package in the form you have received it.
- **[voluntary:]** Let the documentation of your distribution and/or your additional material also reproduce the content of the *notice text file*, a hint to the software name, a link to its homepage, and a link to the Apache 2.0 license.

prohibits to institute any patent litigation against anyone alleging that the software constitutes patent infringement.

²⁰⁸⁾ For details see pp. 61 - 65

²⁰⁹⁾ In this case it doesn't matter whether you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit

²¹⁰⁾ For details see pp. 61 - 64

²¹¹⁾ For implementing the handover of files correctly → OSLiC 67

²¹²⁾ The Apache license seems willingly to be a bit ambigüe: it uses the term ““Notice” text file”. In its strict sense, the term refers to a file named 'NOTICE.[txt|pdf|. . .]'. In a weaker sense, it may denote any (text) file containing (licensing) notices. For being sure to act according to this requirement you should also read this term in the broader sense if there is no text file name 'NOTICE'

6.2.3 ApL-3: Passing the unmodified software as binaries

means that you are going to distribute an unmodified version of the received Apache software to 3rd parties in form of a set of binary files or an integrated binary package²¹³

covers OSUC-02, OSUC-05, OSUC-07²¹⁴

requires the following tasks in order to fulfill the license conditions

- **[mandatory:]** Give the recipient a copy of the Apache 2.0 license. If it is still not incorporated into your binary package, add it²¹⁵.
- **[mandatory:]** Ensure that the licensing elements – esp. the specific copyright notice of the original author(s) – are retained in your package in the form you have received them. If you compile the binary from the sources, ensure that all the licensing elements are also incorporated into the package.
- **[mandatory:]** Ensure that the *notice text file* is retained or integrated into your binary package in the form you have initially received it.
- **[mandatory:]** Ensure that the *notice text file* is also reproduced if and wherever such third-party notices normally appear – especially, if you are distributing an unmodified Apache licensed library as embedded component of your own work which displays its own copyright notice.
- **[voluntary:]** Let the documentation of your distribution and/or your additional material also reproduce the content of the *notice text file*, a hint to the software name, a link to its homepage, and a link to the Apache 2.0 license – especially as subsection of your own copyright notice.

prohibits to institute any patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.4 ApL-4: Passing a modified program as source code

means that you are going to distribute a modified version of the received Apache licensed program, application, or server (proapse) to 3rd parties in form of a set of source code files or an integrated source code package.

covers OSUC-04²¹⁶

²¹³⁾ In this case it doesn't matter whether you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit

²¹⁴⁾ For details see pp. 61 - 64

²¹⁵⁾ For implementing the handover of files correctly → OSLiC 67

²¹⁶⁾ For details see pp. 62

6 Open Source License Compliance: To-Do Lists

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Give the recipient a copy of the Apache 2.0 license. If it is still not incorporated into the source code package, add it²¹⁷.
- **[mandatory:]** Ensure that the licensing elements – esp. the specific copyright notice of the original author(s) – are retained in your package in the form you have received them.
- **[mandatory:]** Ensure that the *notice text file* contains at least all the information of that *notice text file* you have received.
- **[mandatory:]** Ensure that the *notice text file* is also reproduced if and wherever such third-party notices normally appear. If the program already displays a copyright dialog, update it in an appropriate manner.
- **[voluntary:]** Inside of the source code, mark all your modifications thoroughly. Generate a *notice text file*, if it still does not exist. Expand (sic!) the *notice text file* by a description of your modifications.
- **[voluntary:]** Let the documentation of your distribution and/or your additional material also reproduce the content of the *notice text file*, a hint to the software name, a link to its homepage, and a link to the Apache 2.0 license.

prohibits to institute any patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.5 ApL-5: Passing a modified program as binary

means that you are going to distribute a modified version of the received Apache licensed program, application, or server (proapse) to 3rd parties in form of a set of binary files or an integrated binary package.

covers OSUC-04²¹⁸

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Give the recipient a copy of the Apache 2.0 license. If it is still not incorporated into your binary package, add it²¹⁹.
- **[mandatory:]** Ensure that the licensing elements – esp. the specific copyright notice of the original author(s) – are retained in your package in the form you have received them. If you compile the binary from

²¹⁷⁾ For implementing the handover of files correctly → OSLiC 67

²¹⁸⁾ For details see pp. 62

²¹⁹⁾ For implementing the handover of files correctly → OSLiC 67

6 Open Source License Compliance: To-Do Lists

the sources, ensure that all the licensing elements are also incorporated into the package.

- **[mandatory:]** Ensure that the *notice text file* contains at least all the information of that *notice text file* you have received.
- **[mandatory:]** Ensure that the *notice text file* is also reproduced if and wherever such third-party notices normally appear. If the program already displays a copyright dialog, update it in an appropriate manner.
- **[voluntary:]** Inside of the source code, mark all your modifications thoroughly. Generate a *notice text file*, if it still does not exist. Expand (sic!) the *notice text file* by a description of your modifications.
- **[voluntary:]** Let the documentation of your distribution and/or your additional material also reproduce the content of the *notice text file*, a hint to the software name, a link to its homepage, and a link to the Apache 2.0 license – especially as a subsection of your own copyright notice.

prohibits to institute any patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.6 ApL-6: Passing a modified library as independent source code

means that you are going to distribute a modified version of the received Apache licensed code snippet, module, library, or plugin (snimoli) to 3rd parties in form of a set of source code files or an integrated source code package, but without embedding it into another larger software unit.

covers OSUC-08²²⁰

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Give the recipient a copy of the Apache 2.0 license. If it is still not incorporated into the software package, add it²²¹.
- **[mandatory:]** Ensure that the licensing elements – esp. the specific copyright notice of the original author(s) – are retained in your package in the form you have received them.
- **[mandatory:]** Ensure that the *notice text file* contains at least all the information of that *notice text file* you have received.

²²⁰) For details see pp. 65

²²¹) For implementing the handover of files correctly → OSLiC 67

- **[voluntary:]** Inside of the source code, mark all your modifications thoroughly. Generate a *notice text file*, if it still does not exist. Expand (sic!) the *notice text file* by a description of your modifications.
- **[voluntary:]** Let the documentation of your distribution and/or your additional material also reproduce the content of the *notice text file*, a hint to the software name, a link to its homepage, and a link to the Apache 2.0 license.

prohibits to institute any patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.7 ApL-7: Passing a modified library as independent binary

means that you are going to distribute a modified version of the received Apache licensed code snippet, module, library, or plugin (snimoli) to 3rd parties in form of a set of binary files or an integrated binary package but without embedding it into another larger software unit.

covers OSUC-08²²²

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Give the recipient a copy of the Apache 2.0 license. If it is still not incorporated into your binary package, add it²²³.
- **[mandatory:]** Ensure that the licensing elements – esp. the specific copyright notice of the original author(s) – are retained in your package in the form you have received them. If you compile the binary from the sources, ensure that all the licensing elements are also incorporated into the package.
- **[mandatory:]** Ensure that the *notice text file* contains at least all the information of that *notice text file* you have received.
- **[voluntary:]** Inside of the source code, mark all your modifications thoroughly. Generate a *notice text file*, if it still does not exist. Expand (sic!) the *notice text file* by a description of your modifications.
- **[voluntary:]** Let the documentation of your distribution and/or your additional material also reproduce the content of the *notice text file*, a hint to the software name, a link to its homepage, and a link to the Apache 2.0 license – especially as a subsection of your own copyright notice.

²²²) For details see pp. 65

²²³) For implementing the handover of files correctly → OSLiC 67

prohibits to institute any patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.8 ApL-8: Passing a modified library as embedded source code

means that you are going to distribute a modified version of the received Apache licensed code snippet, module, library, or plugin (snimoli) to 3rd parties in form of a set of source code files or an integrated source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10²²⁴

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Give the recipient a copy of the Apache 2.0 license. If it is still not incorporated into the software package, add it²²⁵.
- **[mandatory:]** Ensure that the licensing elements – esp. the specific copyright notice of the original author(s) – are retained in your package in the form you have received them.
- **[mandatory:]** Ensure that the *notice text file* contains at least all the information of that *notice text file* you have received.
- **[mandatory:]** Ensure that the *notice text file* is also reproduced if and wherever such third-party notices normally appear. If your overarching program displays an own copyright dialog, insert these information.
- **[voluntary:]** Inside of the source code, mark all your modifications thoroughly. Generate a *notice text file*, if it still does not exist. Expand (sic!) the *notice text file* by a description of your modifications.
- **[voluntary:]** Let the documentation of your distribution and/or your additional material also reproduce the content of the *notice text file*, a hint to the software name, a link to its homepage, and a link to the Apache 2.0 license.
- **[voluntary:]** Arrange your source code distribution so that the integrated Apache license and the *notice text file* clearly refer only to the embedded library and do not disturb the licensing of your own overarching work. It's a good tradition to keep the embedded components like libraries, modules, snippets, or plugins in specific directory which contains also all additional licensing elements.

²²⁴) For details see pp. 66

²²⁵) For implementing the handover of files correctly → OSLiC 67

prohibits to institute any patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.9 ApL-9: Passing a modified library as embedded binary

means that you are going to distribute a modified version of the received Apache licensed code snippet, module, library, or plugin to 3rd parties in form of a set of binary files or an integrated binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10²²⁶

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Give the recipient a copy of the Apache 2.0 license. If it is still not incorporated into your binary package, add it²²⁷.
- **[mandatory:]** Ensure that the licensing elements – esp. the specific copyright notice of the original author(s) – are retained in your package in the form you have received them. If you compile the binary from the sources, ensure that all the licensing elements are also incorporated into the package.
- **[mandatory:]** Ensure that the *notice text file* contains at least all the information of that *notice text file* you have received.
- **[mandatory:]** Ensure that the *notice text file* is also reproduced if and wherever such third-party notices normally appear. If your overarching program displays an own copyright dialog, insert these information.
- **[voluntary:]** Inside of the source code, mark all your modifications thoroughly. Generate a *notice text file*, if it still does not exist. Expand (sic!) the *notice text file* by a description of your modifications.
- **[voluntary:]** Let the documentation of your distribution and/or your additional material also reproduce the content of the *notice text file*, a hint to the software name, a link to its homepage, and a link to the Apache 2.0 license – especially as subsection of your own copyright notice.
- **[voluntary:]** Arrange your binary distribution so that the integrated Apache license and the *notice text file* clearly refer only to the embedded library and do not disturb the licensing of your own overarching

²²⁶) For details see pp. 66

²²⁷) For implementing the handover of files correctly → OSLiC 67

6 Open Source License Compliance: To-Do Lists

work. It's a good tradition to keep the libraries, modules, snippet, or plugins in specific directories which contain also all licensing elements.

prohibits to institute any patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.10 Discussions and Explanations [tbd]

6.3 BSD licensed software

As an approved open source license, the BSD license exists in two versions²²⁸. The latest release is the *BSD 2-Clause license*²²⁹, the elder release is the *BSD 3-Clause license*²³⁰. The very little differences between the two versions have to be respected exactly. Nevertheless, we could integrate the requirements into one to-do list per use case.

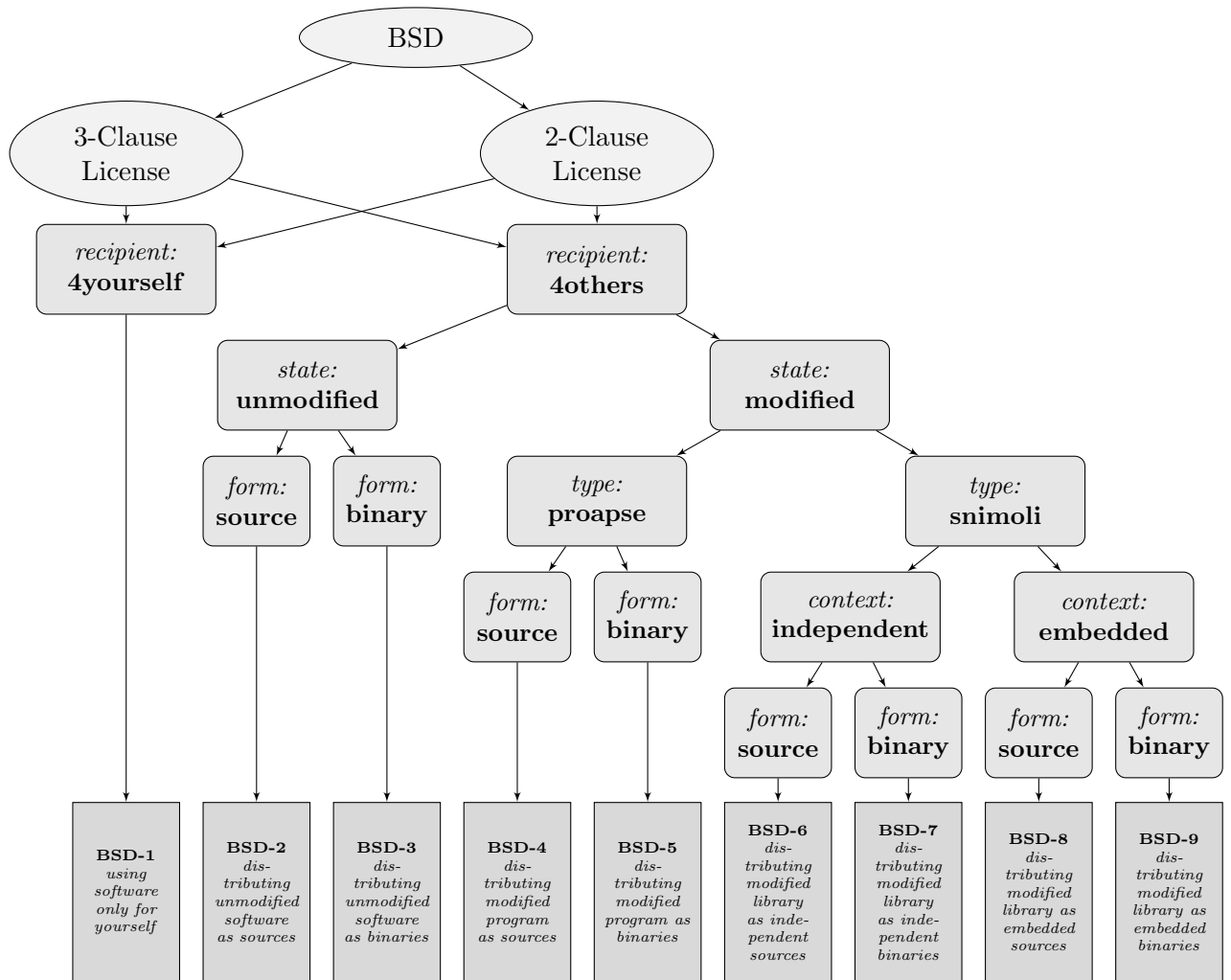
Explicitly, all BSD open source licenses 'only' focus on the (re-)distribution *open source use cases* which we have specified by our token *4others*. Conditions for the other use cases specified by the token *4yourself* can be derived²³¹. Additionally the BSD licenses consider the form of the distribution, esp. whether the work is distributed as a (set of) source code file(s) or as a (set of) the binary file(s). Use the following tree to find the BSD license fulfilling to-do lists.

²²⁸) Following the OSI, there is another 'ancient' BSD license – containing a fourth clause known as advertising clause – which “(...) officially was rescinded by the Director of the Office of Technology Licensing of the University of California on July 22nd, 1999”. Cf. *Open Source Initiative: The BSD 3-Clause License*, 2012, wp. Because of that cancellation you can simply act according the *BSD 3-Clause license* if you have to fulfill the eldest BSD license.

²²⁹) cf. *Open Source Initiative: The BSD 2-Clause License*, 2012, wp.

²³⁰) cf. *Open Source Initiative: The BSD 3-Clause License*, 2012, wp.

²³¹) For details of the *open source use case tokens* see p. 52. For details of the *open source use cases* based on these token see p. 55



6.3.1 BSD-1: Using the software only for yourself

means that you are going to use a received BSD software only for yourself and that you do not handover it to any 3rd party in any sense.

covers OSUC-01, OSUC-03, OSUC-06, and OSUC-09²³²

requires no tasks in order to fulfill the conditions of the BSD license with respect to this use case:

- You are allowed to use any kind of BSD software in any sense and in any context without any obligations premised you do not handover the software to 3rd parties.

prohibits nothing explicitly.

²³²) For details see pp. 61 - 65

6.3.2 BSD-2: Passing the unmodified software as source code

means that you are going to distribute an unmodified version of the received BSD software to 3rd parties in form of a set of source code files or an integrated source code package²³³

covers OSUC-02, OSUC-05, OSUC-07²³⁴

requires the following tasks in order to fulfill the license conditions

- **[mandatorily:]** Ensure that the licensing elements – esp. the BSD license text, the specific copyright notice of the original author(s), and the BSD disclaimer – are retained in your package in the form you have received them.
- **[voluntarily:]** Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.

prohibits nothing explicitly premised you are using the *BSD 2 Clause License*. In contrast, the *BSD 3 Clause License* explicitly prohibits to use the name of the licensing organization or the names of the licensing distributors to promote your own work.

6.3.3 BSD-3: Passing the unmodified software as binary

means that you are going to distribute an unmodified version of the BSD received software to 3rd parties in form of a set of binary files or an integrated binary package²³⁵

covers OSUC-02, OSUC-05, OSUC-07²³⁶

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that your distribution contains the original copyright notice, the BSD license, and the BSD disclaimer in the form you have received them. If you compile the binary file on the base of the source code package and if this compilation does not also generate and integrate the licensing files then create the copyright notice, the BSD conditions, and the BSD disclaimer according to the form of

²³³⁾ In this case it doesn't matter whether you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit

²³⁴⁾ For details see pp. 61 - 64

²³⁵⁾ In this case it doesn't matter whether you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit

²³⁶⁾ For details see pp. 61 - 64

6 Open Source License Compliance: To-Do Lists

the source code package and insert these files into your distribution manually²³⁷.

- **[mandatory:]** Ensure that the documentation of your distribution and/or your additional material also contain the author specific copyright notice, the BSD conditions, and the BSD disclaimer.

prohibits nothing explicitly premised you are using the *BSD 2 Clause License*. In contrast, the *BSD 3 Clause License* explicitly prohibits to use the name of the licensing organization or the names of the licensing distributors to promote your own work.

6.3.4 BSD-4: Passing a modified program as source code

means that you are going to distribute a modified version of the received BSD program, application, or server (proapse) to 3rd parties in form of a set of source code files or an integrated source code package.

covers OSUC-04²³⁸

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that the licensing elements – esp. the BSD license text, the specific copyright notice of the original author(s), and the BSD disclaimer – are retained in your package in the form you have received them.
- **[voluntary:]** Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.
- **[voluntary:]** It is a good practice of the open source community, to let the copyright notice which is shown by the running program also state that the program is licensed under the BSD license. Because you are already modifying the program you can also add such a hint if the presented original copyright notice lacks such a statement.

prohibits nothing explicitly premised you are using the *BSD 2 Clause License*. In contrast, the *BSD 3 Clause License* explicitly prohibits to use the name of the licensing organization or the names of the licensing distributors to promote your own work.

²³⁷⁾ For implementing the handover of files correctly → OSLiC 67

²³⁸⁾ For details see pp. 62

6.3.5 BSD-5: Passing a modified program as binary

means that you are going to distribute a modified version of the received BSD program, application, or server (proapse) to 3rd parties in form of a set of binary files or an integrated binary package.

covers OSUC-04²³⁹

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that your distribution contains the original copyright notice, the BSD license, and the BSD disclaimer in the form you have received them. If you compile the binary file on the base of the source code package and if this compilation does not also generate and integrate the licensing files then create the copyright notice, the BSD conditions, and the BSD disclaimer according to the form of the source code package and insert these files into your distribution manually²⁴⁰.
- **[mandatory:]** Ensure that the documentation of your distribution and/or your additional material also contain the author specific copyright notice, the BSD conditions, and the BSD disclaimer.
- **[voluntary:]** It is a good practice of the open source community, to let the copyright notice which is shown by the running program also state that the program is licensed under the BSD license. Because you are already modifying the program you can also add such a hint if the presented original copyright notice lacks such a statement.

prohibits nothing explicitly premised you are using the *BSD 2 Clause License*. In contrast, the *BSD 3 Clause License* explicitly prohibits to use the name of the licensing organization or the names of the licensing distributors to promote your own work.

6.3.6 BSD-6: Passing a modified library as independent source code

means that you are going to distribute a modified version of the received BSD code snippet, module, library, or plugin (snimoli) to 3rd parties in form of a set of source code files or an integrated source code package, but without embedding it into another larger software unit.

covers OSUC-08²⁴¹

requires the tasks in order to fulfill the license conditions

²³⁹) For details see pp. 62

²⁴⁰) For implementing the handover of files correctly → OSLiC 67

²⁴¹) For details see pp. 65

- **[mandatory:]** Ensure that the licensing elements – esp. the BSD license text, the specific copyright notice of the original author(s), and the BSD disclaimer – are retained in your package in the form you have received them.
- **[voluntary:]** Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.

prohibits nothing explicitly premised you are using the *BSD 2 Clause License*. In contrast, the *BSD 3 Clause License* explicitly prohibits to use the name of the licensing organization or the names of the licensing distributors to promote your own work.

6.3.7 BSD-7: Passing a modified library as independent binary

means that you are going to distribute a modified version of the received BSD code snippet, module, library, or plugin (snimoli) to 3rd parties in form of a set of binary files or an integrated binary package but without embedding it into another larger software unit.

covers OSUC-08²⁴²

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that your distribution contains the original copyright notice, the BSD license, and the BSD disclaimer in the form you have received them. If you compile the binary file on the base of the source code package and if this compilation does not also generate and integrate the licensing files, then create the copyright notice, the BSD conditions, and the BSD disclaimer according to the form of the source code package and insert these files into your distribution manually²⁴³.
- **[mandatory:]** Ensure that the documentation of your distribution and/or your additional material also contain the author specific copyright notice, the BSD conditions, and the BSD disclaimer.

prohibits nothing explicitly premised you are using the *BSD 2 Clause License*. In contrast, the *BSD 3 Clause License* explicitly prohibits to use the name of the licensing organization or the names of the licensing distributors to promote your own work.

²⁴²⁾ For details see pp. 65

²⁴³⁾ For implementing the handover of files correctly → OSLiC 67

6.3.8 BSD-8: Passing a modified library as embedded source code

means that you are going to distribute a modified version of the received BSD code snippet, module, library, or plugin (snimoli) to 3rd parties in form of a set of source code files or an integrated source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10²⁴⁴

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that the licensing elements – esp. the BSD license text, the specific copyright notice of the original author(s), and the BSD disclaimer – are retained in your package in the form you have received them.
- **[voluntary:]** Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.
- **[voluntary:]** It is a good practice of the open source community, to let the copyright notice which is shown by the running program also state that it contains components licensed under the BSD license. Because you are embedding this snimoli into a larger software unit, you are developing this larger unit. Hence, you can also expand the copyright notice of this larger unit by such a hint to its BSD components.
- **[voluntary:]** Arrange your source code distribution so that the the licensing elements – esp. the BSD license text, the specific copyright notice of the original author(s), and the BSD disclaimer – clearly refer only to the embedded library and do not disturb the licensing of your own overarching work. It's a good tradition to keep the embedded components like libraries, modules, snippets, or plugins in specific directory which contains also all additional licensing elements.

prohibits nothing explicitly premised you are using the *BSD 2 Clause License*. In contrast, the *BSD 3 Clause License* explicitly prohibits to use the name of the licensing organization or the names of the licensing distributors to promote your own work.

6.3.9 BSD-9: Passing a modified library as embedded binary

means that you are going to distribute a modified version of the received BSD code snippet, module, library, or plugin to 3rd parties in form of a set of

²⁴⁴) For details see pp. 66

6 Open Source License Compliance: To-Do Lists

binary files or an integrated binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10²⁴⁵

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that your distribution contains the original copyright notice, the BSD license, and the BSD disclaimer in the form you have received them. If you compile the binary file on the base of the source code package and if this compilation does not also generate and integrate the licensing files, then create the copyright notice, the BSD conditions, and the BSD disclaimer according to the form of the source code package and insert these files into your distribution manually²⁴⁶.
- **[mandatory:]** Ensure that the documentation of your distribution and/or your additional material also contain the author specific copyright notice, the BSD conditions, and the BSD disclaimer.
- **[voluntary:]** It is a good practice of the open source community, to let the copyright notice which is shown by the running program also state that it contains components licensed under the BSD license. Because you are embedding this snimoli into a larger software unit, you are developing this larger unit. Hence, you can also expand the copyright notice of this larger unit by such a hint to its BSD components.
- **[voluntary:]** Arrange your binary distribution so that the licensing elements – esp. the BSD license text, the specific copyright notice of the original author(s), and the BSD disclaimer – clearly refer only to the embedded library and do not disturb the licensing of your own overarching work. It's a good tradition to keep the librabries, modules, snippet, or plugins in specific directiers which contain also all licensing elements.

prohibits nothing explicitly premised you are using the *BSD 2 Clause License*. In contrast, the *BSD 3 Clause License* explicitly prohibits to use the name of the licensing organization or the names of the licensing distributors to promote your own work.

²⁴⁵⁾ For details see pp. 66

²⁴⁶⁾ For implementing the handover of files correctly → OSLiC 67

6.3.10 Discussions and Explanations

The *BSD 2-Clause license* has a simple structure: In the beginning, it generally “(permits) [the] redistribution and [the] use in source and binary forms, with or without modification, [...]”, if one fulfills the two rules of the license²⁴⁷. The first rule concerns the (re)distribution in form of source code, the second the (re)distribution of binary packages. Here are some explanations why we translated the rules into which sets of executable tasks:

- For the “redistribution of source code” the license requires, that the package must “[...] retain the above copyright notice, this list of conditions and the following disclaimer”²⁴⁸. Hence, you are not allowed, to modify any of the copyright notes which are already embedded in the received (source) files. And from a logical point of view, there must exist an explicit or implicit assertion that the software is licensed under the *BSD 2-Clause license*²⁴⁹. This is often implemented by simply adding a copy of the license into the package. Hence, you are furthermore not allowed to modify these files or corresponding text snippets. For our purposes, we translated the bans into the following executable task:

Ensure that the licensing elements – esp. the BSD license text, the specific copyright notice of the original author(s), and the BSD disclaimer – are retained in your package in the form you have received them.

- For the redistribution in form of binary files, the license requires, that the licensing elements must be “[...] (reproduced) in the documentation and/or other materials provided with the distribution”²⁵⁰. Hence, this is not required as a necessary condition for the (re)distribution as source code package. But nevertheless, even for a distribution in form of source code, it is often possible to fulfill this rule too – e.g. if you offer an own download site for source code packages. In such cases, it is a sign of respect, to mention the licensing not only inside of the packages, but also in the text of your site. Because of that, we added the following voluntary task for all BSD

²⁴⁷) cf. *Open Source Initiative*: The BSD 2-Clause License, 2012, wp.

²⁴⁸) cf. id., ibid.

²⁴⁹) The BSD license requires that a re-distributed software package must contain the (package specific) copyright notice, the (license specific) conditions and the BSD disclaimer. (cf. id., l.c., wp.) You might ask what you should do, if these elements are missed in the package you received. If so, the package you received had not been licensed adequately. Hence, you do not know reliably whether you have received it under a BSD license. In other words: If you have received a BSD licensed software package, it must contain sufficient license fulfilling elements, or it is not a BSD licensed software.

²⁵⁰) cf. id., l.c., wp.

6 Open Source License Compliance: To-Do Lists

open source use cases which deal with the redistribution in form of source code:

Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.

- Naturally, because the reproduction of the licensing elements “in the documentation and/or other materials provided with the distribution” is explicitly required for the “redistribution in binary form”²⁵¹, we had to rewrite the facultative task for a distribution in form of source code as a mandatory task for all BSD open source use cases which deals with the redistribution in binary form:

Ensure that the documentation of your distribution and/or your additional material also contains the author specific copyright notice, the BSD conditions, and the BSD disclaimer.

- In case of (re)distributing the program in form of binary files, it is sometimes not enough, to pass the licensing elements as one has received them. If you compile the binary package from the source code, it is not necessarily true, that the licensing elements are also automatically generated and embedded into the ‘binary package’. But nevertheless, you have to add the copyright notice, the conditions and the disclaimer to this package for acting according to the BSD license. Therefore we chose the following form of an executable, license fulfilling task for all binary oriented distributions:

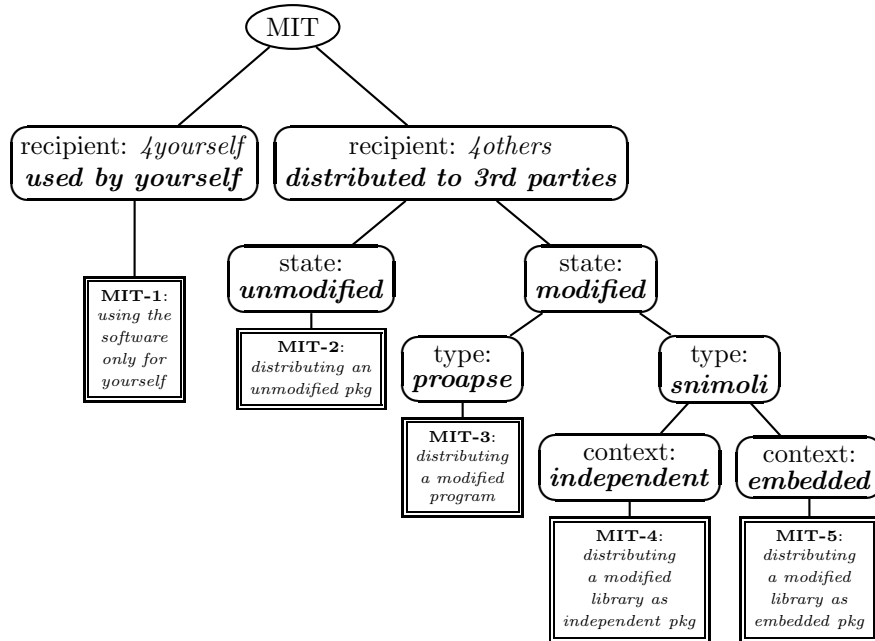
Ensure that your distribution contains the original copyright notice, the BSD license, and the BSD disclaimer in the form you have received them. If you compile the binary file on the base of the source code package and if this compilation does not also generate and integrate the licensing files, then create the copyright notice the BSD conditions, and the BSD disclaimer according to the form of the source code package and insert these files into your distribution manually.

- Finally, we wished to insert a hint to the general (open source) tradition, to mention the used open source software and their licenses as a remark of the ‘copyright widget’ of an application. This is not required by the BSD license. But it is a general, good tradition. Naturally, because of the freedom to use and modify open source software and to redistribute a modified version of it, you are also allowed to insert such references, even if they are missing. Therefore we added a third voluntary license tradition fulfilling task for all relevant open source use cases.

²⁵¹) cf. *Open Source Initiative*: The BSD 2-Clause License, 2012, wp.

6.4 MIT licensed software

The MIT license is known as one of the most permissive licenses. Thus, the MIT specific finder can be simplified:



6.4.1 MIT-1: Using the software only for yourself

means that you are going to use a received MIT software only for yourself and that you do not handover it to any 3rd party in any sense.

covers OSUC-01, OSUC-03, OSUC-06, and OSUC-09²⁵²

requires the tasks in order to fulfill the conditions of the MIT license:

- You are allowed to use any kind of MIT licensed software in any sense and in any context without any other obligations if you do not handover the software to 3rd parties and if you do not modify the existing copyright notes and the existing permission notice.

prohibits nothing explicitly.

6.4.2 MIT-2: Passing the unmodified software

means that you are going to distribute an unmodified version of the received MIT software to 3rd parties – regardless whether you distribute it in form

²⁵²) For details see pp. 61 - 65

of binaries or of source code files²⁵³

covers OSUC-02, OSUC-05, OSUC-07²⁵⁴

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that the licensing elements – esp. the MIT license text containing the specific copyright notices of the original author(s), the permission notices and the MIT disclaimer – are retained in your package in the form you have received them.
- **[voluntary:]** It's a good tradition to let the documentation of your distribution and/or your additional material also contain a link to the original software (project) and its homepage.

prohibits nothing explicitly.

6.4.3 MIT-3: Passing a modified program

means that you are going to distribute a modified version of the received MIT program, application, or server (proapse) to 3rd parties²⁵⁵.

covers OSUC-04²⁵⁶

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that the original licensing elements – esp. the MIT license text containing the specific copyright notices of the original author(s), the permission notices and the MIT disclaimer – are retained in your package in the form you have received them.
- **[voluntary:]** Mark your modifications in the sourcecode, regardless whether you want to distribute the code or not.
- **[voluntary:]** It's a good tradition to let the documentation of your distribution and/or your additional material also contain a link to the original software (project) and its homepage.
- **[voluntary:]** You are allowed to expand an existing copyright notice presented by the program during its user interaction by a hint to your own work or part.
- **[voluntary:]** It is a good practice of the open source community, to let the copyright notice which is shown by the program also state that it is

²⁵³⁾ In this case it also doesn't matter whether you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent package

²⁵⁴⁾ For details see pp. 61 - 64

²⁵⁵⁾ In this case it doesn't matter whether you are going to distribute it in form of a set of source code files or as an integrated source code package.

²⁵⁶⁾ For details see pp. 62

based on a version originally licensed under the MIT license. Because you are already modifying the program, you can also add such a hint, if the presented original copyright notice lacks such a statement.

prohibits nothing explicitly.

6.4.4 MIT-4: Passing a modified library independently

means that you are going to distribute a modified version of the received MIT code snippet, module, library, or plugin (snimoli) to 3rd parties without embedding it into another larger software unit.

covers OSUC-08²⁵⁷

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that the original licensing elements – esp. the MIT license text containing the specific copyright notices of the original author(s), the permission notices and the MIT disclaimer – are retained in your package in the form you have received them.
- **[voluntary:]** Mark your modifications in the sourcecode, regardless whether you want to distribute this source code or not.
- **[voluntary:]** It's a good tradition to let the documentation of your distribution and/or your additional material also contain a link to the original software (project) and its homepage.

prohibits nothing explicitly.

6.4.5 MIT-5: Passing a modified library as embedded component

means that you are going to distribute a modified version of the received MIT code snippet, module, library, or plugin (snimoli) to 3rd parties together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10²⁵⁸

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that the original licensing elements – esp. the MIT license text containing the specific copyright notices of the original author(s), the permission notices and the MIT disclaimer – are retained in your package in the form you have received them.

²⁵⁷⁾ For details see pp. 65

²⁵⁸⁾ For details see pp. 66

6 Open Source License Compliance: To-Do Lists

- **[voluntary:]** Mark your modifications in the sourcecode, regardless whether you want to distribute this source code or not.
- **[voluntary:]** It is a good practice of the open source community, to let the copyright notice which is shown by the running program also state that the program uses a component being licensed under the MIT license. And it is a good tradition to insert links to the homepage / download page of this used component.
- **[voluntary:]** It's also a good tradition to let the documentation of your program and/or your additional material also mention that you have used this component added by a link to the original software component and its homepage.
- **[voluntary:]** Arrange your distribution so that the the original licensing elements – esp. the MIT license text containing the specific copyright notices of the original author(s), the permission notices and the MIT disclaimer – clearly refer only to the embedded library and do not disturb the licensing of your own overarching work. It's a good tradition to keep the libraries, modules, snippet, or plugins in specific directiers which contain also all licensing elements.

prohibits nothing explicitly.

6.4.6 Discussions and Explanations

The MIT-License is known as one of the most permissive licenses. It is a very short license containing (1) a paragraph saying that you are allowed to do almost anything you want, followed (2) by the condition that you have to “include” the existing copyright notes and the permission notes “[...] in all copies or substantial portions of the software”, and (3) closed by the well known disclaimer²⁵⁹. But the license doesn't talk about the difference of source code and object code. So, you have to find the right way by yourself. Here are our interpretations:

- If you do not modify the received MIT licensed application, neither for your own purposes, nor for handing over the program to 3rd parties, you can conclude that all copyright notices and permission notices are already correct.
- Nevertheless, we added the hint not to modify these licensing elements in the context of the use case *used by yourself*. This is evoked by the MIT license itself. It requires explicitly that “ the above copyright notice and this permission notice shall be included in all[sic!] copies or substantial portions of the Software”²⁶⁰ – thus also into those copies you make for your

²⁵⁹) cf. *Open Source Initiative*: The MIT License, 2012, wp.

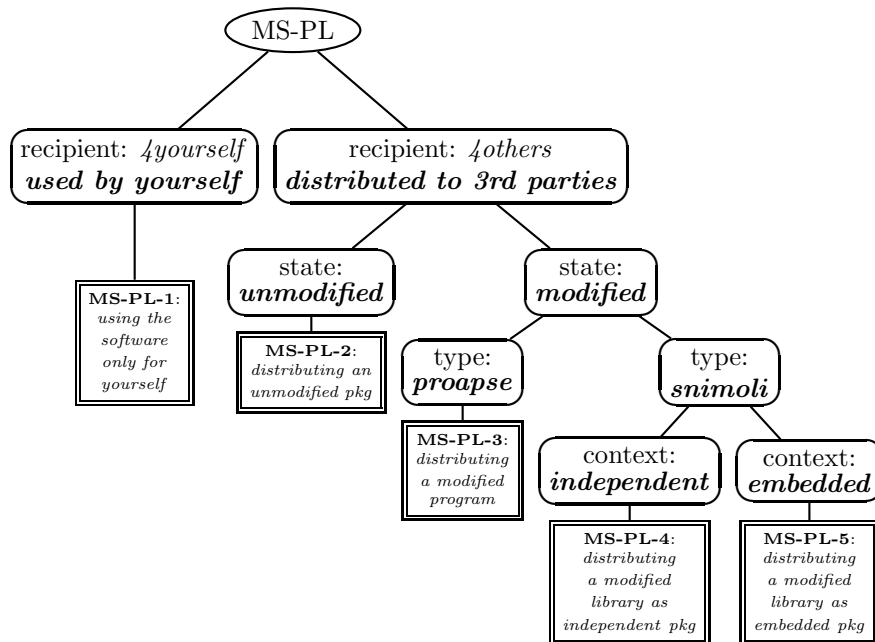
²⁶⁰) cf. id., *ibid*.

own purposes own your own machines, and even if this is probably not so often reviewed.

- If you modify the received MIT licensed application, regardless for which purposes, you are simply not allowed to erase or modify existing copyright notes and permission notices. You may add your own modifications under new conditions, but the old base must survive.

6.5 Microsoft Public License

The MS-PL license is also one of the most permissive licenses. Thus, the MS-PL specific finder can be simplified:



6.5.1 MS-PL-1: Using the software only for yourself

means that you are going to use a received MS-PL software only for yourself and that you do not handover it to any 3rd party in any sense.

covers OSUC-01, OSUC-03, OSUC-06, and OSUC-09²⁶¹

requires the tasks in order to fulfill the conditions of the MS-PL license:

- You are allowed to use any kind of MS-PL licensed software in any sense and in any context without any other obligations if you do not handover the software to 3rd parties.

²⁶¹⁾ For details see pp. 61 - 65

prohibits nothing explicitly.

6.5.2 MS-PL-2: Passing the unmodified software

means that you are going to distribute an unmodified version of the received MS-PL software to 3rd parties – regardless whether you distribute it in form of binaries or of source code files²⁶²

covers OSUC-02, OSUC-05, OSUC-07²⁶³

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that all licensing elements – esp. all copyright, patent, trademark, and attribution notices that are present in the version you received – are completely retained in your package.
- **[mandatory:]** Incorporate a complete copy of the MS-PL license into your package, regardless whether you distribute a source code or a binary package²⁶⁴.
- **[voluntary:]** It's a good tradition to let the documentation of your distribution and/or your additional material also contain a link to the original software (project) and its homepage.

prohibits to use any contributors' name, logo, or trademarks (without an additional or general legally based approval).

6.5.3 MS-PL-3: Passing a modified program as source code

means that you are going to distribute a modified version of the received MS-PL licensed program, application, or server (proapse) to 3rd parties as a source code package.

covers OSUC-04²⁶⁵

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that all licensing elements – esp. all copyright, patent, trademark, and attribution notices that are present in the version you received – are completely retained in your package.
- **[mandatory:]** Incorporate a complete copy of the MS-PL license into your package.

²⁶²⁾ In this case it also doesn't matter whether you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent package

²⁶³⁾ For details see pp. 61 - 64

²⁶⁴⁾ → OSLiC, p. 96

²⁶⁵⁾ For details see pp. 62

- **[mandatory:]** If you do not want to publish your modifications under the MS-PL too, then cleanly separate your own sources and licensing documents from original elements of the adopted work.
- **[voluntary:]** Mark your modifications in the sourcecode.
- **[voluntary:]** It's a good tradition to let the documentation of your distribution and/or your additional material also contain a link to the original software (project) and its homepage (as far as this does not clashes with the prohibitions stated below).
- **[voluntary:]** You are allowed to expand an existing copyright notice of the program by a hint to your own contributions.
- **[voluntary:]** It is a good practice of the open source community, to let the copyright notice which is shown by the running program also state that the program is licensed under the MS-PL license (as far as this does not clashes with the prohibitions stated below). Because you are already modifying the program, you can also add such a hint, if the original copyright notice lacks such a statement.

prohibits to use any contributors' name, logo, or trademarks (without an additional or general legally based approval).

6.5.4 MS-PL-3: Passing a modified program as binary

means that you are going to distribute a modified version of the received MS-PL licensed program, application, or server (proapse) to 3rd parties as a package of binaries.

covers OSUC-04²⁶⁶

requires the tasks in order to fulfill the license conditions

- **[voluntary:]** Mark your modifications in the source code even if do not want to distribute it.
- **[voluntary:]** It's a good tradition to let the documentation of your distribution and/or your additional material also contain a link to the original software (project) and its homepage (as far as this does not clashes with with the prohibitions stated below).
- **[voluntary:]** It is a good practice of the open source community, to let the copyright notice which is shown by the running program also state that the derivative work is based on a version originally licensed under the MS-PL license (as far as this does not clashes with the prohibitions

²⁶⁶) For details see pp. 62

stated below) – perhaps by linking to the project homepage of the original. Because you are already modifying the program, you can also add such a hint, if the original copyright notice lacks such a statement.

prohibits to use any contributors' name, logo, or trademarks (without an additional or general legally based approval)

6.5.5 MS-PL-4: Passing a modified library independently as source code

means that you are going to distribute a modified version of the received MS-PL code snippet, module, library, or plugin (snimoli) to 3rd parties without embedding it into another larger software unit.

covers OSUC-08²⁶⁷

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that all licensing elements – esp. all copyright, patent, trademark, and attribution notices that are present in the version you received – are completely retained in your package.
- **[mandatory:]** Incorporate a complete copy of the MS-PL license into your package.
- **[mandatory:]** If you do not want to publish your modifications under the MS-PL too, then cleanly separate your own sources and licensing documents from original elements of the adopted part(s).
- **[voluntary:]** Mark your modifications in the sourcecode.
- **[voluntary:]** It's a good tradition to let the documentation of your distribution and/or your additional material also contain a link to the original software (project) and its homepage (as far as this does not clashes with with the prohibitions stated below).

prohibits to use any contributors' name, logo, or trademarks (without an additional or general legally based approval)

6.5.6 MS-PL-4: Passing a modified library independently as binary

means that you are going to distribute a modified version of the received MS-PL code snippet, module, library, or plugin (snimoli) to 3rd parties without embedding it into another larger software unit.

covers OSUC-08²⁶⁸

²⁶⁷⁾ For details see pp. 65

²⁶⁸⁾ For details see pp. 65

requires the tasks in order to fulfill the license conditions

- **[voluntary:]** Mark your modifications in the source code even if do not want to distribute it.
- **[voluntary:]** It's a good tradition to let the documentation of your distribution and/or your additional material also contain a link to the original software (project) and its homepage (as far as this does not clashes with with the prohibitions stated below).

prohibits to use any contributors' name, logo, or trademarks (without an additional or general legally based approval)

6.5.7 MS-PL-5: Passing a modified library as embedded source code

means that you are going to distribute a modified version of the received MS-PL licensed code snippet, module, library, or plugin (snimoli) to 3rd parties in form of a set of source code files or an integrated source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10²⁶⁹

requires the tasks in order to fulfill the license conditions

- **[mandatory:]** Ensure that all licensing elements – esp. all copyright, patent, trademark, and attribution notices that are present in the version you received – are completely retained in your package.
- **[mandatory:]** Incorporate a complete copy of the MS-PL license into your package.
- **[mandatory:]** If you do not want to publish your modifications and/or your overarching application under the MS-PL too, then cleanly separate your own sources and licensing documents from original elements of the adopted work.
- **[voluntary:]** Mark your modifications in the sourcecode.
- **[voluntary:]** It's a good tradition to let the documentation of your distribution and/or your additional material also contain a link to the original software (project) and its homepage (as far as this does not clashes with with the prohibitions stated below).
- **[voluntary:]** It is a good practice of the open source community, to let the copyright notice shown by your overarching program also state that it is based on a component originally licensed under the MS-PL

²⁶⁹⁾ For details see pp. 66

6 Open Source License Compliance: To-Do Lists

license – perhaps by linking the project homepage of the original (as far as this does not clashes with the prohibitions stated below).

prohibits to use any contributors’ name, logo, or trademarks (without an additional or general legally based approval)

6.5.8 MS-PL-5: Passing a modified library as embedded binary

means that you are going to distribute a modified version of the received MS-PL licensed code snippet, module, library, or plugin (snimoli) to 3rd parties in form of a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10²⁷⁰

requires the tasks in order to fulfill the license conditions

- **[voluntary:]** Mark your modifications in the source code even if do not want to distribute it.
- **[voluntary:]** It’s a good tradition to let the documentation of your distribution and/or your additional material also contain a link to the original software (project) and its homepage (as far as this does not clashes with with the prohibitions stated below).
- **[voluntary:]** It is a good practice of the open source community, to let the copyright notice shown by your own overarching program also state that it is based on a component originally licensed under the MS-PL license – perhaps by linking the project homepage of the original (as far as this does not clashes with the prohibitions stated below).

prohibits to use any contributors’ name, logo, or trademarks (without an additional or general legally based approval)

6.5.9 Discussions and Explanations

The MS-PL is also a very permissive and short license. It requires to do: (a) You must preserve existing licensing elements. (b) You must distribute the source code as whole or “portions” of the source code under the MS-PL. (c) You must add a copy of the license if you distribute (parts of) the source code. (d) If you distribute a binary package, you must distribute (the parts of) the work under a license “that complies with this (MS-PL) license”²⁷¹.

²⁷⁰⁾ For details see pp. 66

²⁷¹⁾ cf. *Open Source Initiative*: MS-PL, 2013, wp.

The most confusing clause is probably the condition, to “[...] distribute any portion of the software in compiled or object code form [...] only [...] under a license that complies with this license”. But a closer examination is lighting the situation: The only other conditions of the license which refer to the context of distributing binaries are the requirements a) not to abuse trademarks, b) not to bring a patent claim against any contributor, and c) not to expect any warranties or guarantees with respect to the distributed portion²⁷².

Based on these readings we decided ...

- ...to let you incorporate a copy of the license into your distribution even if it only contains the binaries of the unmodified version: if you have not modified it, you do not lose any advantage if you add the license, too. So, this is the best method to fulfill the *MSL-PL binary condition*.
- ...to erase all mandatory conditions in case of the binary distributions: the patent restriction of the MS-PL itself is already covered by the MS-PL patent section of the OSLiC and the no warranty clause of the MS-PL by the OSLiC section concerning the power of the MS-PL while the trademark restrictions are explicitly added into the prohibition section.
- ...to erase the hints to a voluntarily updated copyright dialog in case of distributing a snimoli independently because the copyright dialog normally is designed by the overarching work which uses the library, not by the library itself.

6.6 Postgres Licensed Software in the usage context of ... [tbd]

6.6.1 PGL specific use case 1

(covers OSUC-X - OSUC-Z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]

²⁷²⁾ cf. *Open Source Initiative*: MS-PL, 2013, wp. §3A, §3B, §3E.

- [indirectly:]

6.6.2 PGL specific use case n

(covers OSUC-x - OSUC-z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

6.7 PHP Licensed Software in the usage context of ... [tbd]

6.7.1 PHP specific use case 1

(covers OSUC-X - OSUC-Z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

6.7.2 PHP specific use case n

(covers OSUC-x - OSUC-z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

6.8 Eclipse Licensed Software in the usage context of ... [tbd]

6.8.1 EPL specific use case 1

(covers OSUC-X - OSUC-Z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

6.8.2 EPL specific use case n

(covers OSUC-x - OSUC-z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

6.9 European Union Public Licensed Software in the usage context of ... [tbd]

6.9.1 EUPL specific use case 1

(covers OSUC-X - OSUC-Z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

6.9.2 EUPL specific use case n

(covers OSUC-x - OSUC-z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

6.10 Mozilla Public Licensed Software in the usage context of ... [tbd]

6.10.1 MPL specific use case 1

(covers OSUC-X - OSUC-Z)

6.10.2 MPL specific use case n

(covers OSUC-x - OSUC-z)

6.11 LGPL Licensed Software in the usage context of ... [tbd]

6.11.1 LGPL specific use case 1

(covers OSUC-X - OSUC-Z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

6.11.2 LGPL specific use case n

(covers OSUC-x - OSUC-z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

6.12 AGPL Licensed Software in the usage context of ... [tbd]

6.12.1 AGPL specific use case 1

(covers OSUC-X - OSUC-Z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

6.12.2 AGPL specific use case n

(covers OSUC-x - OSUC-z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

6.13 GPL Licensed Software in the usage context of ... [tbd]

6.13.1 GPL specific use case 1

(covers OSUC-X - OSUC-Z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

6.13.2 GPL specific use case n

(covers OSUC-x - OSUC-z)

means ...

covers OSUC-?? ...

requires the following tasks in order to fulfill the license conditions

- [mandatorily:] Ensure ...
- [voluntarily:] Let ...

prohibits the following doings in order to fulfill the license conditions

- [directly:]
- [indirectly:]

7 Open Source Licenses and Their Legal Environments [tbd]

In this chapter we analyze why to know a license alone is not enough. At the end you will know that open source licenses are embedded into the legal environment of a state. And you will know in which sense the German legal environment predetermines your readings of open source licenses.

8 Conclusion

This chapter shortly describes what the OSLiC is, how it should be used, and how it can be read. It shall be written as top-down explanation.

9 Appendices

9.1 Some Additional Remarks on the OSLiC Quotation Style

We have already characterized the general tone of our footnotes²⁷³. Let us now briefly explain a little peculiarity of our bibliography:

Modern times have also changed the humanities. Formerly a book or an article must be printed for being ripe to be quoted. Our statements relied on static, readily prepared works. Nowadays even university libraries sometimes offer those books and articles as PDF files which are printed in the original. As a scholar, now you must rely on the equality of the printed version and the PDF file - at least with respect to the page numbers and the appearance. You can not verify the equivalence - at least to a certain degree.

Moreover: in case of such 'e-books' and 'e-articles' the libraries often do not offer the pdf files themselves but links to the download pages of the publisher. Formerly as a scholar you could trust that your readers would be able to retrieve the quoted work if they want to verify your citations. It's one task of our libraries to hold available our scientific sources. But now they do not buy any longer the books, but the right to download files over the university net. In this case these PDF files are not stored on the serves of the university library. By using the link provided by the publisher each student or each reader downloads his own file - case by case. Therefore - as a scholar - you now have to trust that the publisher, who provides the link, will not change that pdf file that you have cited.

But it gets even worse: While it might be that publishers modify their work secretly (even it is not very likely that they do it), it's a definite feature of the web that its pages are frequently changed. Hence we must ask ourselves: Can we seriously argue on the basis of statements and documents which might disappear? Can we quote such possibly volatile sources? The problem is: we must do it, especially if we write about an internet topic - and even if we want to write a really reliable compendium.

So, what can we do? First, we must confide in our readers, that they either will retrieve our sources or - if they can not find them - that they believe that we really have found and read what we have written and quoted. Second, we store

²⁷³) → p. 10

9 Appendices

all these e-wares²⁷⁴ we read²⁷⁵. And thirdly we should lay open to our readers the different levels of reliableness of our sources. Therefore we use the following markers in our bibliographic data²⁷⁶:

- Print / Copy:- The source is printed and we saw either the printed work really or we get an official copy by our library. Hence you should also be able to get the work in a library, at least in those we used (UB Frankfurt or ULB Darmstadt).
- BibWeb/[PDF/...] :- The source might be printed, but we read only the electronic version (PDF or other type of format), offered by and over the net of our university libraries (UB Frankfurt or ULB Darmstadt).
- FreeWeb/[PDF/...] :- We read the electronic version offered by the free web. In this case we add the url²⁷⁷ and the date when we downloaded / saw the text.

9.2 Some Widespread Open Source Myths

From the viewpoint of an internet student we have to consider that the web offers a mass of rumors concerning the nature of open source software (Licenses). Here are some of the myths²⁷⁸ we met:

open source tries to improve the world ethically :- No, there's a clear ban to exclude persons, groups, purposes. Thus, there is no chance to exclude anyone from using open source software because he is an ethical or moralic malefactor.

²⁷⁴⁾ Take this little word as (new) generalization of 'e-book', 'e-article', 'e-paper' and so on.

²⁷⁵⁾ But because of the copyright we ourselves are naturally not allowed to offer a download link for them or to send a copy of it to those who want to verify our quotes.

²⁷⁶⁾ And another hint: Nowadays sometimes even scientific libraries don't offer exact 'e-copies' of the original. In some cases one can only get html-versions of articles which formerly were printed as part of journals. In these case the scholar has to use sources which lost their original page-numbers. The same can happen to articles of proceedings etc. which are now only offered as autonomous pdf files with an internal paging. If we quote such kind of articles we try to specify the number of the quoted article in the original row of articles, added - if possible - by an internal page number. But naturally we also try to follow the bibliographic data delivered by that organization which distributes these kind of copies.

²⁷⁷⁾ Please note: Long urls often destroy the pleasing appearance of a text because it's difficult to wrap the lines acceptably. Hence we wished to make it easier for LaTeX to do this job. Therefor we sometimes split the urls and inserted blanks. So you have to erase all blanks if you want to verify our urls.

²⁷⁸⁾ At least one time even a scientific legally discussing book is talking about the "myth around open source licenses" - although only as part of the title: cf *Guibault, Lucie a. Ot van Daalen*: Unravelling the Myth around Open Source Licenses. An Anaysis from A Dutch and European Law Perspective; The Hague: T. M. C. Asser Press, 2006 (= IT & Law, [Vol./No.] 8), ISBN 978-90-6704-214-7, pp. 1ff, especially 209ff.

Changed open source software must be re-published :- No, in a double sense!

There are OS licenses which allow the proprietarization of the modified code. And even the LGPL and the GPL, which clearly try to prevent the proprietarization, do not require generally that a modified code must be (re-)published. Only if you give your modified (L)GPL licensed application as binary to anybody, then you have to handover the modified code too.

Modified open source software must be given back to the whole community

:- No. Again, there are OS licenses which allow the proprietarization of the modified code. And even the LGPL and the GPL - which clearly require, that you also publish the modified code, if you give the modified binary to anybody – do not require that you distribute your modification around the world. LGPL and GPL clearly say that you have to hand over the code to those persons which you give the binary. And if you only give your improvement only one person or a group of person, then you must handover your code only to that persons or only to all members of that group.

Published open source software is open for ever :- No, if this myth says that also all future versions will have to be distributed under an open source license. The copyright holder ever holds the copyright. They can change the licence of next release of its software – but only for the following release, not for the current or for former versions. Those releases, which already have been distributed under an open source license, indeed remain open.

Software can either be open source software or proprietary software :- No. The copyright holders themselves can additionally distribute the code under other conditions when ever they want to do it. That's not a question of the licence, but of the copyright.

The opposite of open source software is commercial Software :- No. First, you are also allowed to use the open source software in any commercial purpose. There's only one point which is excluded in OSS: you are not allowed to ask for a licence fee if you distribute 'open source software'. Second, there are many other forms like freeware, public domain software or anything else which is neither open source software nor Commercial Software. It's pointless to take the question of money as a criterion for distinguish open source software and its opposite. Moreover: Proprietary Software as opposite of open source software should be defined ex negativo: all kind of software, which does not fit the OSD is proprietary.

open source software prohibits to earn money :- No, you are allowed to invent each business model you want. There's only one exception: you are not allowed to ask for a licence fee if you distribute 'open source software' [Achtung: sollte eigentlich nur für GPL gelten]. Historically this mistake might be evoked by Debian: The GNU project missed its kernel while the Linux kernel was already distributed as part of collections which also in-

9 Appendices

clude GNU software. Then, in 1983? Ian Murdock was supported by RMS and its FSF to build a really free distribution (Debian) containing GNU software and the Linux kernel. But Ian Murdock states also, that Debian does not want to earn money.

Modifications of open source software must be marked :- No. This is not a defining postulation of the OSD. The OSD allows licenses to require the mark of modifications. But it does not require from all licenses to require the mark modifications for being an open source license.

Modifications of open source software must be marked by your personal data :- No, it is only required to mark modifications so that a reader could distinguish the modifications from the original code. It's required for saving the integrity of the original author. And therefore it is not required as a constitutive criterion by the OSD. It might be that a license additionally requires your name. But that is not a feature of open source software in general. And at least the licenses discussed by us do not require to insert your name.

The open source Definition determines the conditions to use open source software :- No. The *Open Source Definition* determines which licenses are open source licenses, nothing more. The OSD is a set of necessary conditions to be an open source license. It determines the freedom and the responsibilities of a user as a set of more or less abstract rules. But it does not constitute a set of sufficient tasks which a user has to perform for fulfilling any open source license. Open source licenses may differ by instantiating the OSD criteria. So, if you want to know what you have to do to fulfill a license, you have to go back to the real license of that software you are using.

This section outlines reflections by which we initially focused ourselves on the question why we need an OSLiC and how its content and form should be derived from these needs.

9.2.1 Why

Do we need another book about open source? Do *you* need another book about open source software? Let us address this question from the viewpoint of what we already know, what we instinctively believe and what we may have heard. For example you may presume one or more of the following statements are correct. Or you may even have experienced similar perceptions from your peers or managers. Or you have been told they describe 'open source':

- The *Open Source Definition* offers rules to use open source software.
- Modified open source software must be published.

9 Appendices

- Modified open source software must be given back to the community.
- All generations of open source software will remain open for ever.
- Software can either be open source software or proprietary software.
- The opposite of open source software is commercial software.
- open source software prohibits to earn money.
- Modifications of open source software must be marked explicitly.
- Modifiers of open source software must identify themselves.
- When distributing an open source binary it's enough point to a download page to obtain the source code.
- The aim of open source software is to improve the world ethically.
- open source software is viral and infectious.

Do these conceptions sound familiar to you? Unfortunately, whatever we might believe or wish for, these concepts are incorrect. Naturally we will discuss this issue later on. For the moment let us assume they are indeed incorrect²⁷⁹.

So, again: Do *we* need another book about open source software? *We*, that is – in this case and at least initially – the large German company *Deutsche Telekom AG*. Arguing from the perspective of a large company requires not only identifying the common misconceptions, but catering for the unique needs of a large Enterprise. And indeed the very size of the company brings its own problems.

Large companies use more open source software in more varied contexts than small companies. There is an important question that every company should ask: '*Are we sure that we respect all those requirements of open source software we have to respect?*'. But large companies cannot answer this question as easily as small companies: the large number of diverse open source deployments in different contexts mean that case by case governance, a model that may work in small concerns, is far from appropriate for our needs. This leads to wasting both time and money. Further, the chances of success are small: training at least one employee in each software team as an open source software License expert is unrealistic in terms of cost-efficiency and reliability.

Nevertheless even large companies want to and try to fulfill the rules of open source software thoroughly – especially *Deutsche Telekom AG*. When this company realized that the question *Are we sure that we respect all those rules of open source software correctly which we have to respect* could be problematic, it directly asked some of its employees known as open source enthusiasts to establish a service and a process for answering this question.

²⁷⁹⁾ For those who want directly verify our argumentation, we have generated a condensed summary of the arguments and citations. You can find this summary in our appendices.

9 Appendices

So, it is no surprise that we, the initial authors of this *Open Source License Compendium*, were asked by our employer *Deutsche Telekom AG*. Naturally we were proud to work on an open source topic officially. But while we were doing our job we had to ask ourselves if *we* perhaps needed another book on open source. Our answer was *Yes, we do!* Let us shortly explain, why:

First, we already knew that there exists supporting software. These meta-programs take the code of any other application and try to list those open source components being 'covered' by that application²⁸⁰. But we had also already realised that this supporting software did not always match the way we thought the problem should be solved. Second, we recognized fairly quickly that we need a reliable guide. We personally were asked to give the *ok* for projects of our company. We could not answer such requests on the base of '*Oh yes, I read this in the Heise-Ticker a few days ago*' – even if the *Heise-Ticker* had described the situation completely correctly. We ourselves had to be more reliable than this²⁸¹. Naturally we already knew a great deal about open source software. Even so, our knowledge was not as systematic as necessary. We looked for an open source compendium which adequately described what a project or product development team had to do to fulfill the criteria of its open source licenses. We wanted to use that compendium to the basis of our recommendations.

We were very thorough but we did not find what we were looking for. Our 'little' bibliography attest our seriousness. What we found was a lot of information related to individual issues spread over many sources. We did not find answers to our question even in the specific literature. Let us describe three little steps to increase the understanding of the issue:

Without open source licenses there is no open source movement. Nevertheless in dealing with open source licenses, this is sometimes neglected. Take the *Apache Web Server* as an example: No doubt, it is one of the most important pieces of

²⁸⁰⁾ As general examples let us mention Palamida (<http://www.palamida.com/>) and BlackDuck (<http://www.blackducksoftware.com/>).

²⁸¹⁾ But of course, we have to do ourselves the honor of conceding that we – like many many other German open source enthusiasts – love using the *Heise-Ticker* as main IT information source. Unfortunately, its reputation is stil not high enough that its news can directly be cited.

9 Appendices

open source software²⁸² with a specific license²⁸³. Moreover: the success of the open source movement in the commercial world depends directly on the decision of IBM to replace its corresponding own component in the *IBM WebSphere Application Server* with the free *Apache Web Server*²⁸⁴. Meanwhile many companies use the *Apache Web Server* to act as a web provider. Currently the *Apache http server* – as it has to be named correctly – is used more than twice as much as all the other http server software together²⁸⁵. Hence many business models depend on the Apache License. Another aspect is that even the famous *Apache Cookbook*, which explains the installation, the configuration, and the maintainance of an Apache Web Server in details²⁸⁶, does not mention anything about the license which allows for installation, configuration and maintenance. Neither the index lists the word 'license'²⁸⁷, nor the chapters 'Installation'²⁸⁸ or the chapter 'Miscellaneous'²⁸⁹ mentions the license question in a serious way. There's only one short hint as to the advantage of open source software, i.e. that everybody is allowed to install it²⁹⁰. Can you be sure that you are allowed to do what you are doing on the base of such a phrase?

Naturally, the *Apache Cookbook* is not a book for lawyers, it is a book for administrators and developers. They do not want to get bogged down by legalities,

²⁸²) To prove that the *Apache* is really a piece of open source software one must execute a set of steps: First, you have to note, that *Apache* is something like a meta project, covered by the *Apache Software Foundation*, also known as *ASF* (cf. <http://www.apache.org/>, wp.). Thus, you can not directly jump into the *Apache License*. First of all you have to visit the project site (cf. <http://httpd.apache.org/>, wp.) even if at the end its license link leads you back to the general *Apache License sub site* (cf. <http://www.apache.org/licenses/>, wp.) which announces, that "all software produced by The Apache Software Foundation or any of its projects or subjects is licensed according to the terms of the documents listed below". Only now you can use the offered link for switching to the *Apache License*, Version 2.0, if you want to check your rights and duties. But that is difficult. There does not exist any simple list what you have to do for fulfilling the license. Even the faq (cf. <http://httpd.apache.org/docs/2.2/faq/>, wp.) – meanwhile being moved to a wiki – only says that the server "[...] comes with an unrestrictive license" and that you are allowed to put the code on a CD (cf. <http://wiki.apache.org/httpd/FAQ>, wp.). Hence, from the viewpoint of the ASF the license itself shall answer all questions. [Reference download for all urls: 2011-08-31]

²⁸³) cf. *Apache Software Foundation: Apache License, 2.0*, wp..

²⁸⁴) cf. *Moody: Die Software-Rebellen*, 2001, pp.287ff.

²⁸⁵) cf. *Netcraft: August 2011 Web Server Survey*; 2011 (URL: <http://news.netcraft.com/archives/2011/08/05/august-2011-web-server-survey-3.html>) – reference download: 2011-08-31, wp.

²⁸⁶) cf. *Coar, Ken a. Rich Bowen: Apache Kochbuch*; deutsche Übersetzung v. Jochen Wiedmann; Beijing [...]: O'Reilly, 2004, ISBN 3-89721-371-0, et passim.

²⁸⁷) cf. id., l.c., pp.245ff, esp. p. 250.

²⁸⁸) cf. id., l.c., pp.1ff.

²⁸⁹) cf. id., l.c., pp.219ff.

²⁹⁰) cf. id., l.c., pp.1: " ...einer der Vorzüge von open source software besteht darin, dass jedermann die Erlaubnis zur Erzeugung eines eigenen Installationskits hat ".

9 Appendices

they want to set up an Apache Web Server as fast as possible and get down to work. Indeed, the Apache Cookbook offers a good support. But not only as a company you have to ask yourself whether you are really allowed to do what you are doing. Can you find the answer in the *Apache Cookbook*? No. Can you find it in the license itself? Yes, but it is difficult²⁹¹. So again: Can you find your answer in another book, which is *Amazon's* current top recommendation for the search term '*apache server*'²⁹²? Not really: Sascha Kersken's Apache 2.2 Handbook offers a license chapter, but it is only two pages long²⁹³. Moreover, the rights and duties are condensed into just 5 bullet points which taken together do not explain when the software and the license have to be handed over to a customer and when you are allowed to hide your improvements²⁹⁴.

This brings us to the question of what prevents us from using something like a '*general license cookbook*' which explains all the necessary details and which offers quick access to the relevant points:

Of course we also browsed the internet. At least for German speaking people there is an excellent site concerning the topic *open source licenses*. offered by *iffross*, which, loosely translated, means an *Institute for Legal Aspects of the Free and open source software*²⁹⁵, founded in 2000 as a private institute to track the phenomenon 'free software' from the viewpoint of (German) lawyers²⁹⁶. Besides many other aspects this site offers a very well and thoroughly elaborated FAQ²⁹⁷ and a large list of open source licenses and other related licenses: moreover, evidently it is classifying the open source licenses in those 'without copyleft-effect' (BSD), in those with 'strict copyleft-effect' (GPL) and in those with 'restricted copyleft-effect' (LGPL)²⁹⁸.

However, even this excellent site does not fulfill our needs. It does not offer those context specific to-do lists which companies, developers or project managers can use to ensure their open source software is used in a regular manner.

We therefore evaluated that standard book which is listed in the most legal bibliographies²⁹⁹: the book of Jaeger and Metzger which concerns – loosely translated

²⁹¹) And do we really want our developers and maintainers to read the original licenses? Do we really want them to discover that they also have to check the licenses of the used modules?

²⁹²) Tested on <http://www.amazon.de/> at 2011-08-31.

²⁹³) cf. *Kersken, Sasche: Apache 2.2. Das umfassende Handbuch*; 3rd, refreshed a. expanded edition; Bonn: Galileo Press, 2009, ISBN 978-8362-1325-7, pp.111f.

²⁹⁴) cf. id., l.c., p.112.

²⁹⁵) originally: "Institut für Rechtsfragen der Freien und open source software". Main entry point for its site is the URL <http://www.iffross.org/>.

²⁹⁶) cf. *iffross: Ziele, Aufgaben, Geschichte*; 2011 (URL: <http://www.iffross.org/node/16>) – reference download: 2011-09-05, wp.

²⁹⁷) cf. *iffross: FAQ*; 2011 (URL: <http://www.iffross.org/faq-haeufig-gestellte-fragen>) – reference download: 2011-09-05, wp.

²⁹⁸) cf. *iffross: ifross Lizenz-Center*, 2011, wp.

²⁹⁹) at least in that German judicial literature dealing with open source

9 Appendices

– *the judicial framework requirement for open source software*³⁰⁰. Even the most earliest edition of this book already had a clear structure in its chapter ‘copyright’: For each license mentioned (or at least for each license cluster) it offered a subchapter for the rights and a subchapter for the duties³⁰¹ of the software user³⁰². Many other important aspects of the topic *open source* are discussed, too³⁰³.

But we needed more than this. Despite the quality of the book we were certain that we could not hand over this book to our programmers with the recommendation *check your touched licenses and follow the instructions of the relevant subchapters...* This book did not contain simply checkable to-do lists, neither in the first edition³⁰⁴ and in the second edition³⁰⁵ nor in the recently published third edition³⁰⁶. So, how can a company or a developer or a project manager be sure of fulfilling the requirements of the open source licenses sufficiently if he/she does not have a verified list telling him ‘*do this, and in case of that, do that, and finally do also this*’? Why should he himself implicitly become an open source licenses expert who has to extract the necessary steps out of the literature?

While we were searching for an existing open source compendium, we found an article with the title ‘Compendium for the Publication of open source software’³⁰⁷. It aims to be a ‘pragmatic guidebook’ and an ‘assistance’ for ‘publishing software under the conditions of an open source license’³⁰⁸. Moreover, at the end of this article, its authors formulate ambitiously that their ‘guide’ should be carried out, section by section – for getting a legally water tight process of publishing open

³⁰⁰⁾ cf. *Jaeger, Till a. Axel Metzger: Open Source Software. Rechtliche Rahmenbedingungen der Freien Software*; 1st edition. München: Verlag C.H. Beck, 2002, ISBN 3406484026, pp. V – It can not be any surprise that both authors, Mr. Jaeger and Mr. Metzger are members of ifross (cf. <http://www.ifross.org/personen/>, wp.).

³⁰¹⁾ cf. id., l.c., pp. 30ff.

³⁰²⁾ For getting a good survey of the structure and the line of thought see the contents cf. id., l.c., pp. VIIIff.

³⁰³⁾ pars pro toto: have a look at the chapter concerning the liability: cf. id., l.c., pp. 137ff.

³⁰⁴⁾ cf. id., l.c., pp. VIff.

³⁰⁵⁾ cf. *Jaeger, Till a. Axel Metzger: Open Source Software. Rechtliche Rahmenbedingungen der Freien Software*; 2nd edition. München: Verlag C.H. Beck, 2006, ISBN 3406538037, pp. VIIIff.

³⁰⁶⁾ cf. *Jaeger a. Metzger: Open Source Software. Rechtliche Rahmenbedingungen der Freien Software*, 2011, pp. VIIIff. Naturally we use this latest edition for adopting or discussing systematical aspects.

³⁰⁷⁾ approximately translated

³⁰⁸⁾ cf. *Bretschneider, Ulrich, Rainer Glaschick, a. Gernot Gräfe: Ratgeber für die Veröffentlichung von Open-Source-Software durch eine Hochschule*; in: *Michael Asche et al., editors: Open Source. Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen*; Münster, New York, München [... etc.]: Waxmann, 2008 (= POWeR / Patent Offensive Westfalen Ruhr, [Vol./No.] 3), ISBN 978-3-8309-1845-5, pp. 166f (originally: ein “pragmatischer Ratgeber” zur “Veröffentlichung einer Software unter den Rahmenbedingungen einer Open-Source-Lizenz”).

9 Appendices

source software³⁰⁹.

The authors of this article describe something close to what we were looking for. Indeed, the article lists important aspects which have to be taken in consideration if you want to deal open source software correctly: It announces that no obligation exists to publish code either if you embed GPL code into your proprietary code or if you modify the GPL code. It is only if you hand over your binary to other persons that you have to distribute the code too, but only to them and not to the general public³¹⁰. Additionally the articles explains exactly that software – at least in Germany – can only be acknowledged as open source software by transferring the rights to use – the *'Nutzungsrechte'* – to other people, while the copyright itself – the *'Urheberpersönlichkeitsrecht'* – is not transferable and belongs to the author³¹¹. Moreover, besides other aspects the articles briefly and deeply discusses the problem of the No-Warranty-Clauses which are not valid in Germany and which will therefore automatically be replaced by the liability rules for a donation³¹². And last but not least this article actually summarizes the idea of Copyleft and the differences between LGPL and GPL³¹³.

However some gaps remain. The article does not analyze in which cases a University or a company perhaps *must* publish its developments based on open source software. It does not discern between different licenses and conditions. It also does not discuss what Universities or companies, which (re-)use and/or distribute open source software (internally), must do to fulfill the touched open source licenses. And finally this article does not offer the step by step list as promised.

We did, however, feel supported by this article, in two ways. First, it was a well written summary of some main problems. Second, it stated the necessity to have a compendium for being able to establish a legally 'water-tight' process of publishing open source software³¹⁴. We seemed to be justified in our assumptions. But the open source compendium we were looking for had to be more practical, more processable, more distinguishing and more elaborated.

So again: Did we need a new book about open source software? We had looked for a reliable integrated open source compendium. But we found separate pieces of information and – as we know today – some rumors. Our answer was clear: naturally we did not need a new general book about open source. But what was lacking was a description of what responsible developers, project managers or

³⁰⁹⁾ cf. *Bretschneider, Glaschick, a. Gräfe: Ratgeber für die Veröffentlichung von Open-Source-Software durch eine Hochschule*, 2008a, pp. 186 (originally: ein "Ratgeber", der es erlaubt " (...) die zu berücksichtigende Aspekte (strukturiert abzarbeiten) (...) " und einen "rechtlich nicht angreifbaren Veröffentlichungsprozess" zu ermöglichen).

³¹⁰⁾ cf. id., l.c., pp. 170 and 181.

³¹¹⁾ cf. id., l.c., p. 173.

³¹²⁾ cf. id., l.c., p. 177.

³¹³⁾ cf. id., l.c., p. 181.

³¹⁴⁾ cf. id., l.c., p. 186.

9 Appendices

product developers require to fulfill open source licenses. We needed an *Open Source License Compendium*.

At the best such an *Open Source License Compendium* would contain a set of simply to process *'For-Fulfilling-The-License-To-Do-Lists'*. Additionally it should offer an intuitively user-friendly search option for these lists. In any case, it should share developers and project managers the effort of having to become open source license experts. For the other users, it should also clearly explain why one has to do this and not that. Hence a reliable *Open Source License Compendium* should not only list what one has to do, but should offer both, thoroughly verified reliable details and clearly condensed guidance.

Although we did not find such an open source compendium we were familiar with the spirit of the open source community. Hence we followed one of its most simple rules: *'what you miss you must develop on your own'*. Some principles should help us to achieve our targets:

To-do lists as the core, discussions around them : Our work should be split into two parts. As its core we wanted to offer a set of to-do-Lists. Each of these lists should be relevant to one specific open source license and should be clustered by the open source specific use cases. Around this all those aspects of open source software which influence the interpretation of the licenses and the rules core should be precisely characterized. Nevertheless, the users should be able to skip details and go directly to the section they require.

Quotations with thoroughly specified sources : Even if our users should not be obliged to read every part of the compendium they should not be required to believe us. We wanted to be revisable. Because our sources and our conclusions should be easily verifiable, we decided to use the academic citations and list bibliographic data extensively on the basis that our task should be to collect information, not to invent new 'facts'.

Not the internet alone, also books and articles : We wanted to go back to the originals even if the internet was full of more or less modified copies. We wished to get reliable facts and descriptions. Therefore we decided to evaluate not only the internet but also scientific sources – for example – offered by university libraries.

Not clearing out the forest land, but cutting out a swathe : Even if we had to deal with licenses and their legal aspects we did not want to get lost in detailed discussions. It should not be our task to find out whether a specific kind of handling would still be legal or already forbidden. We did not want to fight against the licenses. We did not want to stretch their ambit or to test their boundary. We wished to accept open source licenses as they are: rules written from developers for developers. And even if some parts

9 Appendices

of these licenses would not be valid with respect to a legal system³¹⁵, we wanted to take them as our guideline – at least while they do not violate more general laws³¹⁶. We simply wanted to *find one proven way* to cross the maybe slightly unsure forest of open source licenses. Even if indeed some clauses of the licenses finally were not enforceable against us we wanted to respect them ‘voluntarily’. We wanted to deliver a set of rules which support users and remove the possibility of becoming involved in license disputes with open source developers or the Free Software Foundation.

Take the text seriously : On the other side we wanted to take our license texts as they were. If they lacked anything³¹⁷, we would interpret the open issues in the spirit of the open source idea. But where the text was clear and definite we wanted to take its propositions as a definite decision – even if that meaning stood against well known open source ‘facts’.

Trust the swarm : We did not want to use our own research alone as a basis. We knew that the swarm is ever stronger than a set of some randomly selected experts. Therefore we decided to publish our text as a still unfinished work, starting with an early release 0.2. And then we wanted to invite the community to complete the compendium together with us. We would elaborate our open source compendium as a set of LaTeX- and BibTeX files which could be developed and managed in GIT or any other version control

³¹⁵) And indeed for example for the GPL one can argue in this way: Even if you take the GPL as a contract of the type ‘donation’ respectively “Schenkung”, it is presented in the form of AGBs respectively “Allgemeine Geschäftsbedingungen” and must therefore follow the general AGB rules. ‘Regrettably’ in Germany these general AGB rules do not allow to exclude each type of warranty. If we follow Oberhem, §11 and §12 of the GPL must be invalid in Germany because of these general AGB rules. Moreover, for Oberhem even §5 – the important clause of the GPL by which you can only get the right to use and to distribute GPL software if you respect the rules of the GPL – seems also to be invalid respectively “unwirksam”. But the good message is that the GPL as whole is not invalid even if it contains invalid clauses. *Oberhem, Carolina: Vertrags- und Haftungsfragen beim Vertrieb von Open Source Software; Dissertation; Hamburg: Verlag Dr. Kovač, 2008 (= Recht der Neuen Medien, [Vol./No.] 50), ISBN 978-3-8300-4075-0, pp. 128, 133ff, 150ff, esp. 146, 159.*

³¹⁶) what they clearly do not do!

³¹⁷) The systematical underdetermination of licenses is a problem being also known in the open source respectively Free Software movement. Following the biography of RMS his main judicial counselor Moglen has stated, that “there is uncertainty in every legal process (...)” and that it seemed to be silly to try “(...) to take out all the bugs (...)”. Nevertheless – so Moglen resp. Williams – the goal of Richard Stallman was “the complete opposite”: He tried “(...) to remove uncertainty which is inherently impossible”. But – and that’s the nub of this analysis – Moglen had to follow Stallmann because of RMS character. And he had to summarize their work so, that “(...) the resulting elegance (of the GPL; KR.), the resulting simplicity (of the GPL; KR.) in design almost achieves what it has to achieve”. Hence we are asked to take the license texts themselves seriously. cf. *Williams: Free as in Freedom. Richard Stallman’s Crusade for Free Software, 2002, pp. 177f.*

9 Appendices

system. And finally we would publish our text under a Creative Commons Attribution-Share Alike German 3.0 license, to allow other people to correct us, to help us or even to take our results for their own purposes.

And so we did. Here is the result. Feel free to use it – according to our licensing.

9.2.2 What

Now we can briefly explain how one should be able to use the compendium:

The Same Idea, Different Licenses :- Here you will find background information to help you interpret open source licenses in the sense of the *Free Software movement*³¹⁸, the *open source software movement*³¹⁹, or the GNU-Project³²⁰. We discuss different ways to cluster open source licenses. Finally we present our own taxonomy based on the labels 'protecting the developer', 'protecting the licensed code' and 'protecting the on-top-developments'. If you are familiar with the methods of grouping different open source licenses and particular if you know that you can not authorize your doings on the base of descriptions of such license groups, then it is enough, in order to understand our line of thought, to briefly note our taxonomy and its wording.

The Problem of Derivated Works :- This chapter is important. In the spirit of software developers we try to explain which kinds of programming evoke a derivated work and which not. Our to-do lists will refer to this analysis.

The Problem of Combining Different Licenses :- You should not ignore this chapter. We will explain why and how combining software of different licenses is not as dangerous as it is often told. The results of this chapter influence the structure of our to-do lists.

open source software and Money :- Here we will shortly discuss ways in which money is no problem. If you already know that it is only prohibited to

³¹⁸⁾ At least at this place you are perhaps expecting that we use the logograms FLOSS, F/OSS, F/LOSS, or whatever. As you will read later on the word *Free* is ambiguous and has strained the use of the concept *Free Software*. Later on we will also talk about the invention of the concept *open source* designed as a 'replacement' and acting as a 'splitter'. The mentioned logograms are introduced to re-establish or – at least – to underline the common history and the common center of 'both' movements, whereby the word *Libre* shall resolve the ambiguity of the word *Free*. For a first survey cf. *Wikipedia (en)*: Free and open source software; n.l., 2011 (URL: http://en.wikipedia.org/wiki/Free_and_open_source_software) – reference download: 2011-09-08, wp..

³¹⁹⁾ For another brief and informative introduction cf. *Fogel*: Producing Open Source Software, 2006, pp. 231ff esp. p. 232f..

³²⁰⁾ We ourselves will stay with the concept *open source* because the OSD specifies the scope of our analysis. But we do it with a deep obeisance to Stallmann and the FSF – even if we know that this will not protect us from the thunderbolt of RMS.

require payment for the act of licensing a piece of open source software to second or third parties and if you already know that this is only forbidden by some licenses, and not by all, than you can postpone the reading of this chapter.

The Problem of Implicitly Freeing Patents :- Here we will illuminate some aspects of software patents and how they are handled by some open source licenses. You should know what licenses implicitly do with your patents. But it is not our intention to write a software patent compendium.

Open Source Use Cases as Principle of Classification :- This is an important chapter. We explain our categories 'Use as it is', 'Modify the Code', 'With Redistribution', 'Without Redistribution', 'Isolated Initial Development', 'On-Top-Development': we develop and discuss our taxonomy with respect to the side effects of 'combining different licenses' and 'generating derived works'. This taxonomy will determine the following chapters.

open source licenses: Find Your Specific To-do Lists :- This is a kind of summary which joins the relevant aspects and elaborates the 'finder for your to-do lists'. This is the chapter which you probably will reuse frequently, even if you do not want to read any of our explanations.

open source license Fulfillment: Classified To-do Lists :- This chapter offers all classified to-do lists. The structure of its subchapters will match the structure of our finder and the structure of our taxonomy.

open source licenses and Their Legal Environments :- Here we discuss why using open source software in a regular manner is not only a question of the licenses themselves but of the kind of the surrounding legal system.

Appendices: Some Widespread Open Source Myths :- Here we make good on our promise to explain why all the propositions mentioned at the beginning of this chapter are wrong. You might read this chapter as a special introduction or a reminder epilogue whenever you want to do.

Periodicals, Shortcuts, and Abbreviations

AGPL	GNU Affero General Public License
ApL	Apache License
BISE	Business & Information Systems Engineering [ISSN: 1867-0202]
BSD	Berkeley Software Distribution (License)
[n.abbr.]	Berkeley Technology Law Journal
BWV	Berliner Wissenschafts-Verlag GmbH
[n.abbr.]	Cultural Anthropology [ISSN: 1548-1360]
CiHB	Computers in Human Behavior [ISSN: 0747-5632]
CotACM	Communications of the ACM [ISSN: 0001-0782]
CR	Computer und Recht. Zeitschrift für die Praxis des Rechts der Informationstechnologien
CRi	Computer Law Review international [ISSN: 1610-7608]
[n.abbr.]	Computers & Education [ISSN: 0360-1315]
[n.abbr.]	Cutter IT Journal [ISSN: 1048-5600]
DDT	Drug Discovery Today [ISSN: 1359-6446]
DSS	Decision Support Systems [ISSN: 0167-9236]
[n.abbr.]	Ethics and Information Technology [ISSN: 1388-1957]
E.C.L.R.	European Competition Law Review
EER	European Economic Review [ISSN: 0014-2921]
EPL	Eclipse Public License
et seqq.	and the following ones
EUPL	European Union Public License
GPL	GNU General Public License
[n.abbr.]	Information & Management [ISSN: 0378-7206]
ibid.	ibidem = latin for 'at the same place'
ICC	Industrial and Corporate Change [ISSN: 0960-6491]
id.	idem = latin for 'the same', be it a man, woman or a group...
IEaP	Information Economics and Policy [ISSN: 0167-6245]
[n.abbr.]	IEEE Software [ISSN: 0740-7459]
ifross	Institut für Rechtsfragen der Freien und Open Source Software
[n.abbr.]	International Information and Library Review [ISSN: 1057-2317]
[n.abbr.]	International Journal of Medical Informatics [ISSN: 1386-5056]
[n.abbr.]	interactions [ISSN: 1072-5520]
ISJ	Information Systems Journal [ISSN: 1365-2575]
ITRB	Der IT-Rechtsberater [ISSN: 1617-1527]
JAIS	Journal of the Association for Information Systems [ISSN: 1536-9323]
JCSC	Journal of Computing Sciences in [Small] Colleges [ISSN: 1937-4771]
JISE	Journal of Information Science and Engineering [ISSN: 1016-2364]
JLEO	Journal of Law, Economics, & Organization [ISSN: 1465-7341]
JMIR	Journal of Medical Information Research [ISSN: 1438-8871]
[n.abbr.]	Journal of Academic Librarianship [ISSN: 0099-1333]

9 Appendices

[n.abbr.]	Journal of Comparative Economics [ISSN: 0147-5967]
[n.abbr.]	Journal of Systems and Software [ISSN: 0164-1212]
JSIS	Journal of Strategic Information Systems [ISSN: 0963-8687]
l.c.	loco citato = latin for 'in the place cited'
LGPL	GNU Lesser General Public License
LJ	Linux Journal [ISSN: 1075-3583]
MIT	Massachusetts Institute of Technology (License)
MPL	Mozilla Public License
Ms-PL	Microsoft Public License
n.abbr.	no abbreviation (known)
[n.abbr.]	netWorker [ISSN: 1091-3556]
n.y.	year not stated / no year
n.l.	location not stated / no location
np.	no page numbering
n.st.	not stated
[n.abbr.]	Organization Science [ISSN: 1047-7039]
PgL	Postgres License
PHP	PHP (License)
[n.abbr.]	Queue [ISSN: 1542-7730]
[n.abbr.]	R&D Management [ISSN: 1467-9310]
RP	Research Policy [ISSN: 0048-7333]
SIGCSE Bulletin ...	SIGCSE Bulletin [ISSN: 0097-8418]
SIGCAS	ACM SIGCAS Computers and Society [ISSN: 0095-2737]
SIGMIS Database ..	ACM SIGMIS - The Data Base for Advances in Information Systems [ISSN: 0095-0033]
SIGSOFT SEN	SIGSOFT Software Engineering Notes [ISSN: 0163-5948]
[n.abbr.]	Stanford Law Review [ISSN: 00389765]
[n.abbr.]	Software Quality Journal [ISSN: 0963-9314]
STHV	Science, Technology & Human Values [ISSN: 0162-2439]
ToIT	Transaction on Internet Technology [ISSN: 1533-5399]
ToSEM	Transactions on Software Engineering Methodology [ISSN: 1049-331X]
Ubiquity	Ubiquity - The ACM IT Magazine and Forum [ISSN: 1530-2180]
UB	'Universitätsbibliothek' = library of university X
ULB	'Universitäts- & Landesbibliothek' = library of university and state X
[n.abbr.]	University of Chicago Law Review
[n.abbr.]	University of Illinois Law Review
[n.abbr.]	University of Pittsburgh Law Review
wp.	webpage / webdocument without any internal (page)numbering
ZGE / IPJ	Zeitschrift für geistiges Eigentum [ISSN: 1867-237x]

Bibliography

Apache Software Foundation: Apache License, Version 2.0; 2004, FreeWeb/Html (URL: <http://www.apache.org/licenses/LICENSE-2.0>) – reference download: 2011-08-31

The original text of the apache license. Specifies the 'Does' and the 'Don'ts'.

Apache Software Foundation: Licenses; 2013 [n.y.], FreeWeb/Html (URL: <http://www.apache.org/licenses/>) – reference download: 2013-02-25

Tells the history of the Apache Licenses and marks the releases 1.0 and 1.1 as 'historical' - only release 2.0 has been classified as an open source license by the OSI.

Bretschneider, Ulrich, Rainer Glaschick, a. Gernot Gräfe: Ratgeber für die Veröffentlichung von Open-Source-Software durch eine Hochschule; in: *Michael Asche et al., editors*: Open Source. Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen; Münster, New York, München [... etc.]: Waxmann, 2008 (= POWeR / Patent Offensive Westfalen Ruhr, [Vol./No.] 3), Print, ISBN 978-3-8309-1845-5, pp. 167–188

This article summarizes basic aspects of an Open Source publication succesfully. It addresses the (German) difference between copyright ('Urheberrecht') and transferring the right to use ('Nutzungsrecht'). It mentions the problem of liability. It discusses the choice of an Open Source License with respect to the indented purpose and many things more. Nevertheless the article can't be taken as the sought-after 'Open Source Compendium': it doesn't analyze in which cases a University perhaps must publish its' developments or what it must do for fulfilling the licenses of internally (re-)used and distributed Open Source Software.

Buchtala, Rouven: Determinanten der Open Source Software-Lizenzwahl. Eine spieltheoretische Analyse; Frankfurt am Main, Berlin, Bern [... etc.]: Peter Lang, 2007 (= Informationsmanagement und strategische Unternehmensführung), [Vol./No.] 12), Print, ISBN 978-3-631-57114-9

The book tries to detect why specific OS-licenses are chosen. Game theoretic aspects shall help to answer the question. Thereto OS-licenses are classified as 'permissive', 'restrictive' and 'highly restrictive' licenses. This differentiation highlights some necessary constraints for respecting a license. But the names of these categories are generated with respect to a company which wants to be able to (re)use and (re)sell OS code with minimal restrictions. The intention of the OS licensors to defend the licensed freedom for revocations is not covered by these names.

Coar, Ken a. Rich Bowen: Apache Kochbuch; deutsche Übersetzung v. Jochen Wiedmann; Beijing [...]: O'Reilly, 2004, Print, ISBN 3-89721-371-0

Significantly the 'Apache License' which has enabled the wide use of this http server is not discussed in this book. It doesn't ask wether you are allowed to use the software or under which conditions you may use which module, although these answers are necessary to do really what the book describes.

Debian: The Debian Free Software Guidelines (DFSG); 2013 [n.y.], FreeWeb/HTML (URL: http://www.debian.org/social_contract#guidelines) – reference download: 2013-01-22

The Debian Free Software Guideline contains nine criteria for being free software, These criteria are embedded into the Debian Social Contract and are also adopted by the Open Source Definition.

Bibliography

- Eclipse Foundation*: Eclipse Public License, Version 1.0; 2005 [n.y. of the page itself], FreeWeb/HTML (URL: <http://www.eclipse.org/org/documents/epl-v10.php>) – reference download: 2013-02-20
The Eclipse Public License 1.0, as it is presented by the Eclipse Foundation.
- European Community a. European commission Joinup*: European Union Public Licence v. 1.1. 2007, FreeWeb/HTML (URL: <http://joinup.ec.europa.eu/system/files/EN/EUPL%20v.1.1%20-%20Licence.pdf>) – reference download: 2013-02-08
The English version of the EPL - due to the OSI (<http://opensource.org/licenses/EUPL-1.1>) also the translations have been certified
- European Community a. European commission Joinup*: New FSF statements on the EUPL are a step in the right direction; 2013 [n.y.], FreeWeb/HTML (URL: <https://joinup.ec.europa.eu/community/eupl/news/new-fsf-statements-eupl-are-step-right-direction>) – reference download: 2013-03-05
The answer of the EU to the FSF statement concerning the limited copyleft effect of the EUPL-1.1
- Europäische Gemeinschaft a. European commission Joinup*: Open-Source-Lizenz für die Europäische Union; 2007, FreeWeb/HTML (URL: <http://joinup.ec.europa.eu/system/files/DE/EUPL%20v.1.1%20-%20Lizenz.pdf>) – reference download: 2013-02-08
The German version of the EPL - due to the OSI (<http://opensource.org/licenses/EUPL-1.1>) also the translations have been certified
- Fogel, Karl*: Producing Open Source Software; How to Run a Successful Free Software Project; Beijing, Cambridge, Köln [...]: O'Reilly, 2006, Print, ISBN 978-0-596-00759-1
Well known english written standard work. Focuses on the process of developing Open Source Software. Licences are discussed in a small chapter. Describes very clearly the concepts of 'Free Software', 'Open Source Software', 'FLOSS' and so on. Analyzes also the topic 'License Compatibility'.
- Free Software Foundation*: GNU General Public License, version 2; 1991 [n.y. of the html page itself], FreeWeb/HTML (URL: <http://www.gnu.org/licenses/gpl-2.0.html>) – reference download: 2013-02-05
The GPL-2.0 license, using a strict copyleft. The first release was published in 1989. One of the main authors is probably Richard M. Stallman itself, even if today the Free Software Foundation is the copyright holder.
- Free Software Foundation*: Various Licenses and Comments about Them; 2013 [n.y.], FreeWeb/HTML (URL: <http://www.gnu.org/licenses/license-list.html>) – reference download: 2013-02-08
The GNU list of software licenses, classified by the categories 'GPL-Compatible Free Software Licenses', 'GPL-Incompatible Free Software Licenses', and 'Nonfree Software Licenses'
- Free Software Foundation*: Various Licenses and Comments about Them []; 2013 [n.y.], FreeWeb/HTML (URL: <http://www.gnu.org/licenses/license-list.en.html#EUPL>) – reference download: 2013-03-05
This section states that the 'strong copyleft effect' of the EUPL can be subverted and that therefore it can only offer a weak copyleft effect.
- Guibault, Lucie a. Ot van Daalen*: Unravelling the Myth around Open Source Licenses. An Anaysis from A Dutch and European Law Perspective; The Hague: T. M. C. Asser Press, 2006 (= IT & Law, [Vol./No.] 8), Print, ISBN 978-90-6704-214-7
The book describes the Open Source idea from the viewpoint of the Dutch legal system. It ends in five recommendations for enforcing the clarity of OS licenses and their usage. Except for the title the myth of Open Source is not explicitly mentioned or discussed.
- ifross*: FAQ; 2011, FreeWeb/HTML (URL: <http://www.ifross.org/faq-haeufig-gestellte-fragen>) – reference download: 2011-09-05

Bibliography

This page presents quintessences concerning the topic 'Open Source' - in form of a FAQ list
ifross: Ziele, Aufgaben, Geschichte; 2011, FreeWeb/HTML (URL: <http://www.ifross.org/node/16>) – reference download: 2011-09-05

This page describes the targets and the history of ifross, the 'Institut für Rechtsfragen der Freien und Open Source Software'. It was founded in 2000 as a private institute which shall accompany the phenomenon 'free software' from the viewpoint of (German) lawyers.

ifross: License Center; 2011 [n.y.], FreeWeb/HTML (URL: http://www.ifross.org/ifross_html/lizenzcenter-en.html) – reference download: 2013-02-26

This page lists many more (Open Source) Licenses than the OSI itself. It's classifying the Open Source Licenses in those without copyleft-effect, in those with strict copyleft-effect and in those with restricted copyleft-effect - one of the most sophisticated and elaborated considerations!

Jaeger, Till a. Axel Metzger: Open Source Software. Rechtliche Rahmenbedingungen der Freien Software; 1st edition. München: Verlag C.H. Beck, 2002, Print, ISBN 3406484026

This is the first edition of the most important German standard work concerning Open Source Software Licenses. It deals with many important topics like liability, patents, or branding, And with respect to each of the mentioned licenses (or license clusters) it discusses already the rights and the duties of the software user - but regrettably not in form of a processable todo-list and not with distinguishing the different use cases of the software development process. But nevertheless: it's a very important groundwork!

Jaeger, Till a. Axel Metzger: Open Source Software. Rechtliche Rahmenbedingungen der Freien Software; 2nd edition. München: Verlag C.H. Beck, 2006, Print, ISBN 3406538037

This is the second strongly expanded edition of the most important German standard work concerning Open Source Software Licenses. Like ifross it classifies licenses as those with 'strict copyleft clauses', those with 'restricted copyleft-clauses' and those without any 'copyleft-clauses'. Beside other aspects it now discusses how to achieve acceptance of an Open Source licensing and analyzes the AGB side effects. And again it discusses the rights and the duties of the software user, at least with respect to the most import Open Source Licenses - but again regrettably not in form of a processable todo-list and not with distinguishing the different use cases of the software development process. But naturally: it remains a very important groundwork!

Jaeger, Till a. Axel Metzger: Open Source Software. Rechtliche Rahmenbedingungen der Freien Software; 3rd edition. München: Verlag C.H. Beck, 2011, Print

This is the third current edition of the most important German standard work concerning Open Source Software Licenses. It retains the 'established structure of the chapters', adds new licenses and discusses - beside many other aspects - the compatibility respectively combinability of different licenses. Hence this work again specifies the rights and the duties of the software user, at least with respect to the most important licenses - but again regrettably not in form of a processable todo-list and not with distinguishing the different use cases of the software development process. But it retains the most important groundwork!

Kersken, Sasche: Apache 2.2. Das umfassende Handbuch; 3rd, refreshed a. expanded edition; Bonn: Galileo Press, 2009, Print, ISBN 978-8362-1325-7

An Apache compendium with more than 900 pages offering 2 pages concerning the history and the content of the Apache license.

MLA: MLA Handbook for Writers of Research Papers; 7th edition. New York: The Modern Language Association of America, 2009, Print, ISBN 978-1-60329-024-1

The American writer style, whose merits we know and whom we nevertheless do not follow.

Moody, Glyn: Die Software-Rebellen. Die Erfolgsstory von Linus Torvalds und Linux; transl. from the American [edition, 2000] by Annemarie Pumpering; Landsberg am Lech: verlag moderne industrie, 2001, Print, ISBN 3-478-38730-2

Bibliography

If your friend only wants to read one book on the topic 'Open Source', then give him this! It tells the story - of nearly all aspects.

Moody, Glyn: Rebel Code: Linux And The Open Source Revolution; [New York]: Basic Books, 2002, Print, ISBN ISBN 978-0738206707

The original, last version: If your friend only wants to read one book on the topic 'Open Source', then give him this! It tells the story - of nearly all aspects.

Mozilla Foundation: Mozilla Public License 2.0 (MPL-2.0); 2012, FreeWeb/HTML (URL: <http://www.mozilla.org/MPL/2.0/>) – reference download: 2013-03-05

The Mozilla Public License 2.0 as it is offered by the Mozilla Foundation

Mozilla Foundation: About MPL 2.0: Revision Process and Changes FAQ; 2013 [n.y.], FreeWeb/HTML (URL: <http://www.mozilla.org/MPL/1.1/>) – reference download: 2013-03-05

Differences between the MPL-1.1 and MPL-2.0: The mozilla foundation itself says thatt the most important part of the license - the file-level copyleft - is essentially the same in MPL 2.0 and MPL 1.1.

Mozilla Foundation: Mozilla Public License Version 1.1; 2013 [n.y.], FreeWeb/HTML (URL: <http://www.mozilla.org/MPL/1.1/>) – reference download: 2013-03-05

The Mozilla Public License 1.1 as it is offered by the Mozilla Foundation

Netcraft: August 2011 Web Server Survey; 2011, FreeWeb/Html (URL: <http://news.netcraft.com/archives/2011/08/05/august-2011-web-server-survey-3.html>) – reference download: 2011-08-31

Monthly offered statistic counting sites and their delivering web servers: Following this sheet the apache http server is the most often used software since 1996. And currently it is used more than twice as much as all the others together.

Oberhem, Carolina: Vertrags- und Haftungsfragen beim Vertrieb von Open Source Software; Dissertation; Hamburg: Verlag Dr. Kovač, 2008 (= Recht der Neuen Medien, [Vol./No.] 50), Print, ISBN 978-3-8300-4075-0

The book analyzes the liability of authors and distributors of GPL licensed software. This is necessary because in Germany the NO-Warranty clauses of the GPL are not applicable. They are substituted by the rules of 'gifts'. In this sense private authors and distributors are only liable in case of acting deliberately ["vorsätzlich"] or strongly negligently ["grob fahrlässig"]. Additionally commercial distributors must thouroughly check whether a program might damage a customer. But commercial distributors have the full liability for their additionally offered specific services. Beside the 'AGB' character and the typus of GPL as a contract the book refers the idea of Open Source seriously.

Open Source Initiative: GNU General Public License, version 2 (GPL-2.0). Version 2, June 1991; 1991 [n.y. of the html page itself], FreeWeb/HTML (URL: <http://opensource.org/licenses/GPL-2.0>) – reference download: 2013-02-05

The Gnu Public License 2.0 as it is offered by the OSI

Open Source Initiative: The GNU Lesser General Public License, version 2.1 (LGPL-2.1); 1999 [n.y. of the html page itself], FreeWeb/HTML (URL: <http://opensource.org/licenses/LGPL-2.1>) – reference download: 2013-03-06

The Lesser Gnu Public License 2.1, as it is offered by the OSI

Open Source Initiative: Apache License, Version 2.0; 2004 [n.y. of the page itself], FreeWeb/HTML (URL: <http://opensource.org/licenses/Apache-2.0>) – reference download: 2013-02-07

The Apache 2.0 license, as it is presented by the OSI

Open Source Initiative: Eclipse Public License, Version 1.0; 2005 [n.y. of the page itself], FreeWeb/HTML (URL: <http://opensource.org/licenses/EPL-1.0>) – reference download: 2013-02-20

Bibliography

- The Eclipse Public License 1.0, as it is presented by the OSI*
- Open Source Initiative: European Union Public License, version 1.1 (EURL-1.1; 2007 [n.y. of the html page itself], FreeWeb/HTML (URL: <http://opensource.org/licenses/EURL-1.1>) – reference download: 2013-03-04
- In contrast to most of the other approved open source licenses, the OSI is offering the European Union Public Licences only as a page linked to the multi lingual versions of EURL.*
- Open Source Initiative: GNU General Public License, version 3 (GPL-3.0); 2007 [n.y. of the html page itself], FreeWeb/HTML (URL: <http://opensource.org/licenses/GPL-3.0>) – reference download: 2013-03-05
- The Gnu Public License 3.0 as it is offered by the OSI*
- Open Source Initiative: The GNU Lesser General Public License, version 3.0 (LGPL-3.0); 2007 [n.y. of the html page itself], FreeWeb/HTML (URL: <http://opensource.org/licenses/LGPL-3.0>) – reference download: 2013-03-06
- The Lesser Gnu Public License 3.0, as it is offered by the OSI*
- Open Source Initiative: The BSD 2-Clause License; 2012 [n.y.], FreeWeb/HTML (URL: <http://www.opensource.org/licenses/BSD-2-Clause>) – reference download: 2012-07-03
- The latest current BSD license, known as 2 clause BSD license.*
- Open Source Initiative: The BSD 3-Clause License; 2012 [n.y.], FreeWeb/HTML (URL: <http://www.opensource.org/licenses/BSD-3-Clause>) – reference download: 2012-07-04
- The elder BSD license, known as 3 clause BSD license.*
- Open Source Initiative: The MIT License; 2012 [n.y.], FreeWeb/HTML (URL: <http://opensource.org/licenses/mit-license.php>) – reference download: 2012-08-24
- The MIT license, the most permissive Open Source License*
- Open Source Initiative: The Open Source Definition; 2012 [n.y.], FreeWeb (URL: <http://www.opensource.org/docs/osd>) – reference download: 2012-06-21
- This page lists the ten rules which together specify under which conditions a specific license can be classified as Open Source license. It presents the Open Source Definition. So, a piece of software is only then Open Source Software, if its license has been approved as an Open Source License by the Open Source Initiative.*
- Open Source Initiative: The Open Source Initiative; 2012 [n.y.], FreeWeb (URL: <http://www.opensource.org/about/>) – reference download: 2013-01-22
- By this page the Open Source Initiative describes itself, its' targets and goals.*
- Open Source Initiative: The Open Source Licenses, alphabetically sorted; 2012 [n.y.], FreeWeb (URL: <http://opensource.org/licenses/alphabetical>) – reference download: 2013-01-22
- This page lists the approved Open Source Licenses*
- Open Source Initiative: The [OSI] Licence Review Process; 2012 [n.y.], FreeWeb (URL: <http://www.opensource.org/approval>) – reference download: 2013-01-22
- By this page, the OSI describes its' approval process*
- Open Source Initiative: OSI Mailing List. License-discuss. Draft of new OSI licenses landing page; 2012 [n.y.], FreeWeb/HTML (URL: <http://projects.opensource.org/pipermail/license-discuss/2012-April/000332.html>) – reference download: 2013-01-29
- Thread discussing a new license taxonommy.*
- Open Source Initiative: Microsoft Public License (MS-PL); 2013 [n.y.], FreeWeb/HTML (URL: <http://opensource.org/licenses/MS-PL>) – reference download: 2013-02-26
- The Microsoft Public License, as it is presented by the OSI*
- Open Source Initiative: Mozilla Public License 2.0 (MPL-2.0); 2013 [n.y.], FreeWeb/HTML (URL: <http://opensource.org/licenses/MPL-2.0>) – reference download: 2013-02-07
- The Mozilla Public License 2.0 as it is offered by the OSI*

Bibliography

- Open Source Initiative*: Open Source Licenses by Category; 2013 [n.y.], FreeWeb (URL: <http://opensource.org/licenses/category>) – reference download: 2013-01-29
This page categorizes the approved Open Source Licenses
- Open Source Initiative*: The PHP License 3.0 (PHP-3.0); 2013 [n.y.], FreeWeb/HTML (URL: <http://opensource.org/licenses/PHP-3.0>) – reference download: 2013-02-27
The PHP-3.0 license, as it is presented by the OSI
- Open Source Initiative*: The PostgreSQL Licence (PostgreSQL); 2013 [n.y.], FreeWeb/HTML (URL: <http://opensource.org/licenses/PostgreSQL>) – reference download: 2013-02-27
The PostgreSQL license, as it is presented by the OSI
- Reincke, Karsten*: Classical Scholar Texts With Footnotes based on LaTeX, BibTeX, Koma, jurabib and mykeds-CSR; 2012, FreeWeb/Html (URL: <http://www.fodina.de/en/closedprojects/latex-addons/classical-scholar.html>) – reference download: 2013-02-10
Short description and demonstration of a subtler use of secondary literature
- Reincke, Karsten*: (Geistes-) Wissenschaftliche Texte mit jurabib. Dienst am Leser, Dienst am Scholaren: Über Anmerkungsapparate in Fußnoten - aber richtig. [n.l.], 2012 (URL: <http://download.fodina.de/fodinaClassicalScholarFoNoDe.pdf>) – reference download: 2013-02-10, FreeWeb/PDF
More legitimizing details on a subtler method to use secondary literature adequately.
- Rosen, Lawrence*: Open Source Licensing. Software Freedom and Intellectual Property Law; Upper Saddle River, New Jersey: Prentice Hall PTr, 2005, ISBN 0-13-148787-6
Well-conceived book which discusses different aspects by contrasting the Academic Licenses (also known as permissive Open Source Licenses) and the Reciprocal Licenses (also known as copylefted Open Source Licenses). But one has to be careful: to define the opposite of ASL and OSL by the opposite of reciprocal and not reciprocal licenses does not match the set of OSL in the sense of the OSI.
- Stallman, Richard*: Viewpoint: Why "Open Source" Misses the Point of Free Software; in: Communications of the ACM, 52 June (2009), No. 6, pp.31–33 (URL: <http://doi.acm.org/10.1145/1516046.1516058>) – reference download: 2011-12-29, BibRef/PDF
A newer refreshment of the elder position that open source software as label does not emphasize the idea of freedom: it refers to a development methodology while free software should be seen as a social movement.
- Stallman, Richard M.*: The Danger of Software Patents; transcript of a speech given at University of Cambridge, London on the 25th of March 2002; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, pp.95–111, Print
Stallman emphasizes that a patent cover ideas and the use of ideas while copyrights only cover the details of expression of a work. Then he discussed approaches to react on a patent which tries to influence the development.
- Stallman, Richard M.*: Free Software Definition; originally written in 1996; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, pp.41–43, Print
This article has defined the four constitutive features of free software: the freedom to run a program without any restrictions, the freedom to study how it works, the freedom to redistribute copies of the work and the freedom to improve it and to release the improvements. Additionally the article helps to interpret these features: The accessibility to the sourcecode for example is a necessary condition, not a value itself. Or there might be free software which is not copyleft software.
- Stallman, Richard M.; Gay, Joshua, editor*: Free Software, Free Society: Selected Essays of Richard M. Stallman; [with an] Introduction by Lawrence Lessig; Boston, MA USA: GNU Press, 2002, Print, ISBN 1-882114-98-1
A collection of those important articles by which RMS has established the philosophy of GNU.

Bibliography

Stallman, Richard M.: Free Software: Freedom and Cooperation; transcript of a speech given at New York University on 29 May 2001; In *Stallman: Free Software, Free Society: Selected Essays*, 2002, pp. 155–186, Print

This transcript summarizes the main ideas of Free Software and their genesis: it tells the history of RMS.

Stallman, Richard M.: The GNU Project; originally published in 'Open Sources: Voices from the Open Source Revolution, O'Reilly, 1999'; In *Stallman: Free Software, Free Society: Selected Essays*, 2002, pp. 15–30, Print

This article summarizes aspects of the history, the targets, and the philosophy of GNU.

Stallman, Richard M.: What is Copyleft? originally written in 1996; In *Stallman: Free Software, Free Society: Selected Essays*, 2002, pp. 89–90, Print

This article describes Copyleft as a method for making a program free software and requiring all modified and extended versions of the program to be free software as well: firstly one states that the code is copyrighted, then - as copyright holder - one allows to use, modify, and redistribute the sourcecode under the condition, that this permission is not deleted.

Stallman, Richard M.: Why 'Free Software' is Better than 'Open Software'; originally written in 1998; In *Stallman: Free Software, Free Society: Selected Essays*, 2002, pp. 55–60, Print

This article argues that the concept 'Open Source' was designed not to raise that users deserve freedom. And it underlines that the GNU project sticks to the term 'free software' for spreading the idea of freedom without any fear.

Stallman, Richard M.: Let's Limit the Effect of Software Patents, Since We Can't Eliminate Them; in: *Wired*, n.st. January (2012), p. wp <URL: <http://www.wired.com/opinion/2012/11/richard-stallman-software-patents/>> – reference download: 2013-02-18, FreeWeb/HTML, ISSN n.st.

Tries to solve the challenge of software patents by '[...] (legislating) that developing, distributing, or running a program on generally used computing hardware does not constitute patent infringement'.

Stallman, Richard M.: Fighting Software Patents - Singly and Together; n.st. [2004], FreeWeb/HTML <URL: <http://www.gnu.org/philosophy/fighting-software-patents.html>> – reference download: 2013-02-18

Deals with the fight against the planned new European patent law

Wikipedia (de): Microsoft Public License; n.l, 2013 [n.y.], FreeWeb/HTML <URL: http://de.wikipedia.org/wiki/Microsoft_Public_License> – reference download: 2013-02-26

German explanation of the MS-PL which erroneously says that the Microsoft Public license evokes a weak copyleft effect.

Wikipedia (de): Microsoft Reciprocal License; n.l, 2013 [n.y.], FreeWeb/HTML <URL: <http://de.wikipedia.org/wiki/Ms-RL>> – reference download: 2013-02-26

German explanation of the MS-RL which correctly describe its weak copyleft effect.

Wikipedia (en): Free and open source software; n.l., 2011, FreeWeb/HTML (German Version unter <http://de.wikipedia.org/wiki/FLOSS>) <URL: http://en.wikipedia.org/wiki/Free_and_open_source_software> – reference download: 2011-09-08

Illuminates the attempt to (re-)amalgamate the split Free Software movement and Open Source movement at least on the level of meta-concepts by simply linking the majuskels of Free, Libre, Open, Source, Software as hybrid concepts. 'Libre' is introduced to resolve the ambiguity of 'free' in the sense of 'freedom'.

Wikipedia (en): MIT License; n.l, 2011, FreeWeb/HTML <URL: http://en.wikipedia.org/wiki/MIT_License> – reference download: 2011-09-20

The article explains clearly that there doesn't exist one single MIT-License: one must distinguish between the simpler MIT-elpat-License and the more complex MIT-X11-License. But

Bibliography

basically the two license denote the same license class. They protect the developer, not the code.

Wikipedia (en): Copyleft; n.l., 2013 [n.y.], FreeWeb/HTML (URL: <http://en.wikipedia.org/wiki/Copyleft>) – reference download: 2013-02-02

Describes and deliniates strong and weak copyleft

Wikipedia (en): Permissive free software licence; n.l., 2013 [n.y.], FreeWeb/HTML (URL: http://en.wikipedia.org/wiki/Permissive_free_software_licence) – reference download: 2013-02-02

May be read as an example for the use of the concept 'permissive free software licenses'.

Wikipedia (en): Shared source; n.l., 2013 [n.y.], FreeWeb/HTML (URL: http://en.wikipedia.org/wiki/Shared_source) – reference download: 2013-02-26

Brief summary of the different license models of Microsoft which also explains the difference between the permissive license MS-PL and the weak copyleft license MS-RL.

Williams, Sam: Free as in Freedom. Richard Stallman's Crusade for Free Software; Beijing [... etc.]: O'Reilly, 2002, Print, ISBN 0-596-00287-4

This is the first(?) biography of RMS. It contains so many interviews and background information that it might be read as primary source, not only for the history of RMS, but also for the history of the Free Software movement and its internal counterpart, the Open Source movement.