Version 1.0.3

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[‡]

2018-06-14

†) 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 arbritarily selected experts. We are proud to have its support. Gladly we thank (in alphabetical order):

Eitan Adler,
Stefan Altmeyer (Deutsche Telekom AG),
Ronald Dauster,
John Dobson,
Steffen Härtlein,
Ta'Id Holmes (Deutsche Telekom AG),
Michael Kern (Deutsche Telekom AG),
Michael Machado (Deutsche Telekom AG),
Thorsten Müller (Deutsche Telekom AG),
Tanja Neske (Deutsche Telekom AG),
Oliver Podebradt (Deutsche Telekom AG),
Thomas Quiehl (Deutsche Telekom AG),
Peter Schichl (Deutsche Telekom AG),
Michael Schierl,
Helene Tamer (T-Systems Internationl AG),
Bernhard Tsai (Deutsche Telekom AG),
Thomas Weißschuh (Amadeus Germany GmbH),
. . . additionally all the feedback giving participants of the European Legal & Licensing Workshop 2013 in Amstardam and all the others. . .

Contents

Backlog

- Insert task lists for AL, AFL, CDDL, MPL-1.[0—1], MS-RL, OSL
- Complete the concept of being a derivative work in the context of software development
- Explain how to deal with modifications transforming a proapse into a snimoli and v.v.
- Discuss license compatibility
- Explain the relationship between open source and earning money
- Enrich the literature list

Table 1: History of the Open Source License Compendium

2015-03-01	1.0.0	Target Release ▷ Form expanded by a 6th AGPL relevant question ▷ Expanded OSUC-03 by AGPL subtypes L(ocal) & I(nternet) ▷ Added AGPL specific finder and license fulfilling to-do lists
2015-01-21	0.99.9	▷ added solution for the reverse engineering challenge
2014-03-09	0.99.1	▷ Generate data file for use in OSCAd from the LaTeXsource ▷ Fixed Bug in LGPL C9 Case ▷ general copy-editing of chapter 6
2014-01-08	0.98.2	▷ New section about the patent clauses in the CDDL ▷ hyperlinked PDF file (using hyperref and pdftex) ▷ general copy-editing of chapter 1 to 5
2013-11-27	0.98.1	Korean FLOSS conference release
2013-08-19	0.97.2	▷ incorpation of the typo fixes offered by M.Schierl ▷ some improvements concerning the derivative work ▷ enhancing that the OSLiC deals with prototypic cases
2013-07-28	0.97.1	▷ indirectly used secondary literature added ▷ LGPL specific finder improved ▷ OSCAd aligned, interface improved
2013-05-20	0.96.1	Linux Days release ▷ open source use cases and licenses specific usecase renamed ▷ version matches the content of OSCAd
2013-04-15	0.95.2	FSFE LLW post release ▷ to-do lists for nearly all popular OSI licenses ▷ improved finder for GPL and EUPL ▷ simplified form and improved structure of the OSLiC finder
2013-04-05	0.95.1	FSFE LLW pre release ▷ to-do lists for all permissive and all weak copyleft licenses
2013-03-15	0.94.1	Chemnitzer Linux Day release ▷ to-do lists for all permissive and some weak copyleft licenses ▷ branches merged and new master published
2013-03-08	0.90.1	CeBIT release ▷ to-do lists for the some important licenses added
2013-02-16	0.8.90	▷ 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	▷ 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 ▷ BSD specific mini finder & BSD 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 ▷ added an OSLiC disclaimer & many bibliographic data
2011-09-29	0.1.4	▷ document history 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. At the end it shall deliver reliable information. 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. The OSLiC is only dealing with prototypic use cases. So, it may inspire developers, managers, open source experts, and companies to find good solutions which they finally should let be reviewed by legal counselors.[1]

[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 general.[2] The intention is, rather, for it to be a tool for simplifying the activities for achieving license conformity.

This compendium was created out of necessity at *Deutsche Telekom AG* to counter a challenge some of its software developers and project managers were facing: Of course, the company itself wants to behave as license compliantly as its employees, but, 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 their 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.

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* will prevent the need to increase the size of the OSRB 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 handout,' but to enable the whole community to use, modify, and 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 LaTeXsources,

publicly available via the open repository GitHub.[3] And it licensed the OSLiC under Creative Commons Attribution-ShareAlike 3.0 Germany License.[4]

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 for gathering information concerning the 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 why.[5]

Not clearing the forest, but cutting a swath The OSLiC has to deal with licenses and their legal aspects, no doubt. But it shall not discuss all details of

[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. 313. 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).

every aspect. It shall focus on one possible way to act according to a license in a specific usecase—even if it is known that there might be alternatives.[6]

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 rationally arguing software developers. The OSLiC shall respect that open source licenses are written for software developers (and sometimes by developers).

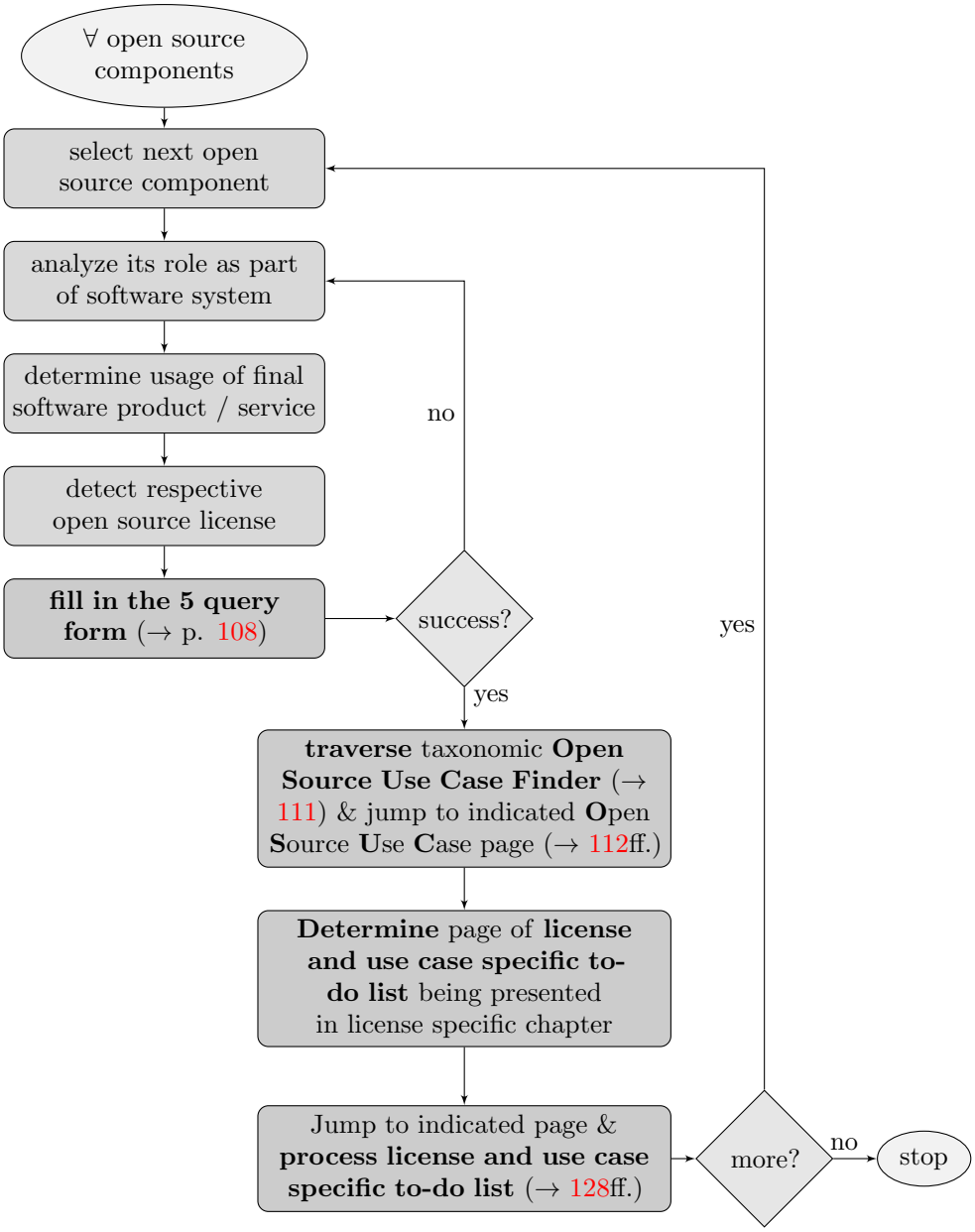
Trust the swarm! The OSLiC shall be open for improvements and adjustments encouraged and stimulated also by other people than employees of *Deutsche Telekom AG*.

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 process:[7]

[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. 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. 108), the taxonomic *Open Source Use Case Finder* (→ 111) or the *O*pen *S*ource *U*se *C*ase page (→ 112ff.): Some of the chapters dedicated to specific open source licenses start 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 being systematically guided by the OSLiC process.



Second, the readers might wish to comprehend the whole analysis. So, we briefly discuss open source license taxonomies as the basis for a license compliant behavior.[8] We consider some side effects of acting according to the open source licenses.[9] Finally, we study the structure of open source use cases.[10]

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] → OSLIC "Open Source: The Same Idea, Different Licenses", pp. 17
[9] → OSLiC "Open Source: About Some Side Effects", pp. 49
[10] → OSLiC "Open Source Use Cases: Concept and Taxonomy", pp. 103

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 prototypical licenses' which are supposed to provide simplified conditions for acting 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, we, too, offer a new way of structuring 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[11] is commonly done. Even the set of the *open source licenses*[12] 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,*[13] maintained by the *Open Source Initiative*[14] and often abbreviated as *OSD*. So, this *OSD* demarcates 'the group of [potential] open source licenses' against 'the group of not open sources licenses.'[15]

Another way to cluster the *Free Software Licenses* is specified by the "Free

[11] Talking about licenses is sometimes a bit tricky: Normally, they have a longer official name and a well known, often abbreviating inofficial nickname. But that's not enough for talking about a specific license adequately: one has additionally to refer to the version of the license itself. The Linux Foundation offers a set of normalized names and identifiers, to minimize the confusion how to denote a license correctly (cf. *The Linux Foundation*: SPDX License List; 2013 ⟨URL: http://spdx.org/licenses/⟩ – reference download: 2014-03-14, wp). The OSLiC tries to use these SPDX identifiers as far as possible. But sometimes the OSLiC wants to group specific licenses by their authors without discriminating the release numbers. Then, the OSLiC uses prefixes of the SPDX.

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

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

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

[15] 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⟩ – reference download: 2013-01-22, wp).

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 [...]"[16] Surprisingly this definition implies that the requirement *the sourcecode 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."[17]

The difference between the OSD and the FSD has often been described as a difference of emphasis:[18] Although both definitions "[...] (cover) almost exactly the same range of software", the *Free Software Foundation*—as it is said—"prefers [...] (to emphazise) the idea of freedom [...]" while the *OSI* wants to underline the philosophically indifferent "development methodology."[19]

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."[20]

---

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

17) cf. id., ibid.

18) 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 imparative" cf. *Stallman, Richard*: Viewpoint: Why "Open Source" Misses the Point of Free Software; in: Commununications of the ACM, 52 June (2009), No. 6 ⟨URL: http://doi.acm.org/10.1145/1516046.1516058⟩ – reference download: 2011-12-29, p. 31

19) 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.

20) 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.

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 licenses.[21]

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 OSD.[22] But only the *OSI approved licenses* have successfully passed the OSI process[23] and therefore are officially listed as *open source licenses*.[24] 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 us 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."[25] But then again, meeting the definition is not enough for being an official open source license: the license has to be approved by the OSI.[26] Thus, it follows that all *OSI approved licenses* are also *DFSG compliant licenses*, but not vice versa.

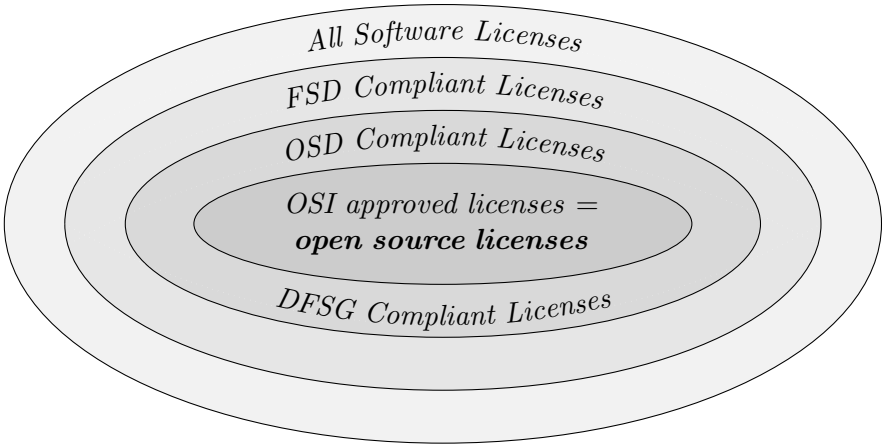
Third, by ignoring the "few exceptions" which have appeared "over the years,"[27] it can be said that, because of their 'kinsmanlike' relation, at least the *OSD compliant licenses* are also *DFSG compliant licenses* and vice versa.

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 it.[28] These conditions are covered by at least the first three paragraphs of the OSD concerning the topics "Free Redistribution," "Source Code," and "Derived Works."[29] 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

[21] Indeed, for analyzing the extensional power of the definition we have to regard all potentially covered licenses, not only the already existing licenses, because the subset of really existing licenses still could be expanded be developing new licenses which fit the definition.
[22] cf. *Open Source Initiative*: The Open Source Definition, 2012, wp.
[23] cf. id., ibid.
[24] cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.
[25] cf. *Fogel*: Producing Open Source Software, 2006, p. 233.
[26] cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.
[27] cf. *Fogel*: Producing Open Source Software, 2006, p. 233.
[28] cf. *Stallman*: Free Software Definition, 1996, p. 41.
[29] cf. *Open Source Initiative*: The Open Source Definition, 2012, wp.

OSD compliant license.[30] 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 OSD.[31] So, the set of all (potential) *Free Software Licenses* must 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 basis of the OSD which does not stipulate 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 different way of grouping the listed licenses would better fit the needs of the visitors of the OSI site.[32] And finally the OSI came up with the categories "popular and widely used (licenses) or with strong communities," "special purpose licenses," "other/miscellaneous licenses," "licenses that are redundant with more popular

[30] For example, see the condition that "the license must be technology-neutral" (cf. *Open Source Initiative*: The Open Source Definition, 2012, wp).

[31] 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 classfied as open source licenses. We are referring to the fact that there might be generated licenses which fulfill the FSD, but not the OSD.

[32] 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.

licenses," "non-reusable licenses," "superseded licenses," "licenses that have been voluntarily retired," and " uncategorized licenses."[33]

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,"[34] the *reciprocal 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,"[35] the *standard licenses,* named as such because they refer to the reusability of "industry standards,"[36] and the *content licenses*, named as such because they refer to "[...] other than software, such as music art, film, literary works" and so on.[37]

Both kinds of taxonomies directly help to find the relevant licenses that should be used for new (software) projects. But again: none of these categories allows us to infer 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 does not determine a license fulfilling behaviour.

Only the act of grouping into *academic licenses* and *reciprocal licenses* touches the idea of license fulfillment tasks, 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."[38] 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) licenses:[39] 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 licenses.[40]

---

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

34) 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.

35) cf. id., l.c., p. 70.

36) cf. id., ibid.

37) cf. id., l.c., p. 71.

38) cf. id., ibid.

39) cf. id., l.c., p. 179.

40) 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).

Similiar in nature to the clustering into *academic licenses* and *reciprocal licenses* is the grouping into *permissive licenses*, *weak copyleft licenses*, and *strong copyleft licenses*: Even Wikipedia 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."[41]

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 also introduce the subset of "strong restrictive licenses" which additionally require that an (overarching) derivative work must be published under the same license.[42] The next refinement of such clustering concepts directly uses the categories "[open source] licenses with a strict copyleft clause,"[43] "[open source] licenses with a restricted copyleft clause,"[44] and "[open source] licenses without any copyleft clause."[45] 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."[46]

Based on this approach to an adequate clustering and labeling,[47] we can develop

[41] 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.

[42] 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.

[43] Originally stated as "Lizenzen mit einer strengen Copyleft-Klausel." Cf. *Jaeger a. Metzger*: Open Source Software. Rechtliche Rahmenbedingungen der Freien Software, 2011, p. 24.

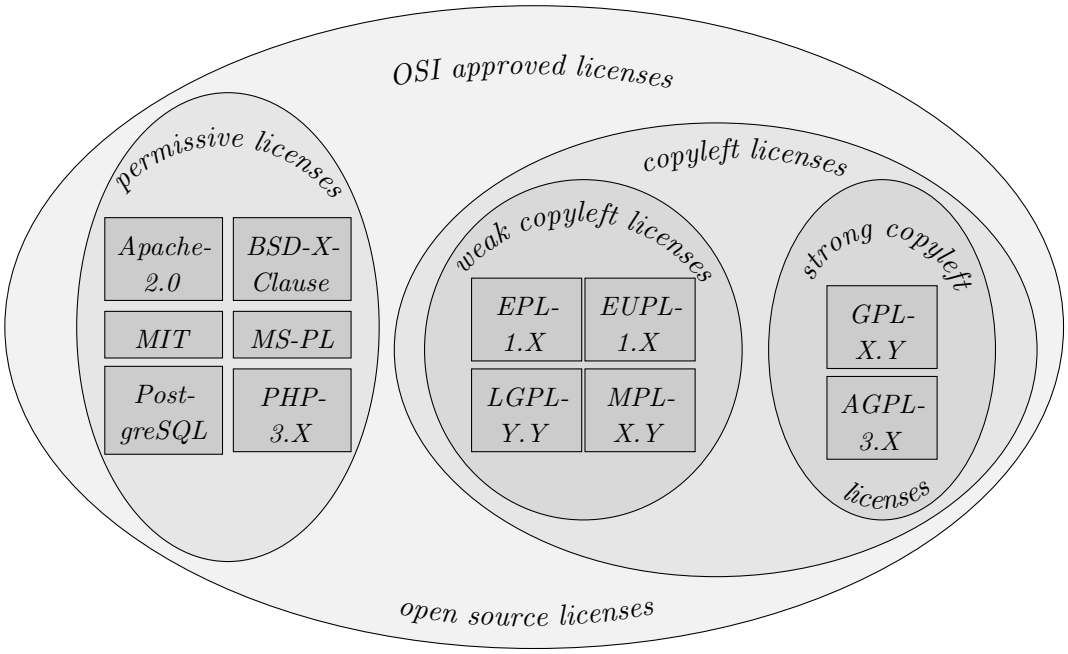
[44] Originally stated as "Lizenzen mit einer beschränkten Copyleft-Klausel." Cf. id., l.c., p. 71.

[45] Originally stated as "Lizenzen ohne Copyleft-Klausel." Cf. id., l.c., p. 83.

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

[47] Finally, we should also mention that there exists still 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 lets 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 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

the following picture:



This extensionally based clarification of a possible open source license taxonomy is probably well-known and often—more or less explicitly—referred to.[48] 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 some sense better than the *copyleft licenses*. Naturally, this 'conclusion' is evoked by confusing the extensional 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.'[49] But there might be other people and companies who emphasize

Open Source Licenses, alphabetically sorted, 2012, wp).

[48] 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')

[49] De facto, *copyleft* is not *copyleft*. Apart from the definition, its effect depends on the particuar licenses which determine the conditions for applying the copyleft 'method.' For example, in the GPL, the copyleft effect is bound to the criteria of 'being distributed.' Later

the protecting effect of the copyleft licenses. And, indeed, at least the open source license[50] *GPL*[51] has initially been developed to protect the freedom, to enable the developers to help their "neighbours", and to get the modifications back:[52] 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."[53] It is a method[54] by which "[...] the code and the freedoms become legally inseparable".[55] 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—prompted 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, second, 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

on, we will collect these conditions systematically (see chapter *Open Source Use Cases: Concept and Taxonomy*, pp. 103). Therefore, here we still permit ourselves to use a somewhat 'generalizing' mode of speaking.

50) Although RMS naturally prefers to call it a *Free Software License* (s. p. 18)

51) 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

52) 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 when the early free community was 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).

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

54) Based on the American legal copyright system, this method uses two steps: first one states, "[...] that it is copyrighted [...]" and second 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.).

55) cf. id., ibid.

as simple as possible.[56] But because of several aspects, this understanding of the 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 imposes other obligations in addtion to republishing derivative works. For example, it also requires giving "[...] any other recipients of the [GPL licensed] Program a copy of this License along with the Program."[57] Furthermore, the 'weakly copylefted' licenses require also more and different criteria to be fulfilled for acting in accordance with 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 charge,[58] 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 [...]"[59] And last but not least, also the *permissive licenses* require tasks to be fulfilled for a license compliant usage—moreover, they also require different things. For example, the BSD license 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 named.[60] 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 files publicly[61]—parts which can contain

[56] And indeed, in the experience of the authors sometimes such simplifications gain their independent existence and determine decisions at 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.

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

[58] 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)

[59] 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.

[60] 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.

[61] 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,

much more information than only the names of the authors or 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 implicit 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*:[62]

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. Moreover, to "protect" the "rights" of the licensees is explicitly mentioned in the GPL-2.0,[63] in the LGPL-2.1,[64] and the GPL-3.0[65]—by which the LGPL-3.0 inherits this purpose.[66] Following this viewpoint, we want to generally assume that open source licenses are designed 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:

---

wp. section 4.4.

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

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

64) 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.

65) 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.

66) 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. prefix.

- 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-privatization of their further development.
- Fourth, we want to assume that new on-top developments being based on open source components can be protected against the privatization for enlarging the world of freely usable software.[67]

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 reference.[68]

2.1 The protecting power of the GNU Affero General Public License (AGPL)

[TODO...]

[67] 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 to long to be used by the following pages

[68] → table 2.1 on p. 28. In February 2014, the Black Duck list of the "Top 20 Open Source Licenses" additionally mentions the Artistic License (AL), the Code Open Project License, the Common Public License, the zlib/png License, the Academic Free License (AFL), the Microsoft Reciprocal License (MS-RL) and the Open Software License (OSL) (cf. *Black Duck*: Top 20 Open Source Licenses; 2014 [n.y] ⟨URL: http://www.blackducksoftware.com/resources/data/top-20-open-source-licenses⟩ – reference download: 2014-02-11, wp.). The Code Open Project License and Common Public License are still not OSI approved open source licenses. (cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.). Thus, finally the OSLiC should additionally analyze not only the AGPL and the CDDL, but also the AL, the AFL, the MS-RL, the OSL and the zlib/png License for being able to justiufiably say, that the OSLiC covers the most important open source licenses.

Table 2.1: Open Source Licenses as Protectors

Open Source Licenses[a]		*are protecting*												
		Users				**Contributors** (Distributors)			**Open Source Software**					**On-Top Develop.**
		who have already got sources or binaries				*who spread open source software*			not distributed	distributed as				
										unmodified		modified		
										sources	binaries	sources	binaries	
		against												
		the loss of the right to			Patent Disputes	Loss of Feedback	Warranty Claims	Patent Disputes	Re-Closings / Re-Privatization of already opened software					Privatization
		use it	modify it	redistribute it										
Apache	2.0	✓	✓	✓	✓	¬	✓	✓	¬	✓	¬	✓	¬	¬
BSD	3-Cl	✓	✓	✓	¬	¬	✓	¬	¬	✓	¬	✓	¬	¬
	2-Cl	✓	✓	✓	¬	¬	✓	¬	¬	✓	¬	✓	¬	¬
MIT		✓	✓	✓	¬	¬	✓	¬	¬	✓	¬	✓	¬	¬
MS-PL		✓	✓	✓	✓	¬	✓	✓	¬	✓	¬	✓	¬	¬
PostgreSQL		✓	✓	✓	¬	¬	✓	¬	¬	✓	¬	✓	¬	¬
PHP	3.0	✓	✓	✓	¬	¬	✓	¬	¬	✓	¬	✓	¬	¬
CDDL	1.0	✓	✓	✓	–	–	–	–	–	–	–	–	–	–
EPL	1.0	✓	✓	✓	✓	✓	✓	✓	¬	✓	✓	✓	✓	¬
EUPL	1.1	✓	✓	✓	✓	✓	✓	✓	¬	✓	✓	✓	✓	¬
LGPL	2.1	✓	✓	✓	¬	✓	✓	¬	¬	✓	✓	✓	✓	¬
	3.0	✓	✓	✓	✓	✓	✓	✓	¬	✓	✓	✓	✓	¬
MPL	1.0	–	–	–	–	–	–	–	–	–	–	–	–	–
	1.1	–	–	–	–	–	–	–	–	–	–	–	–	–
	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. 48) does not cover them

2.2 The protecting power of the Apache License (Apache-2.0)

As an approved *open source license,*[69] the Apache License[70] 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 binaries.[71] Furthermore, based on its patent clause,[72] the Apache-2.0 protects the users against patent disputes.[73] Because of this patent clause and the "disclaimer of warranty" together with the "limitation of liability," the Apache license also protects the contributors and distributors against patent disputes and warranty claims.[74] Finally, the Apache-2.0 protects the distributed sources themselves *against* a change of the license which would *convert* the work *to 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 received.[75]

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 Apache-2.0 does not protect the undistributed software or the distributed binaries against re-closing (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 Apache-2.0 does not protect the on-top developments against privatization.

---

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

70) 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, 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, 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 license Apache-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-2-Clause and BSD-3-Clause licenses.

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

72) → OSLiC pp. 54

73) cf. id., l.c., wp. §3.

74) cf. id., l.c., wp. §3, §7, §8.

75) cf. id., l.c., wp. §4.

2.3 The protecting power of the BSD licenses

As approved *open source licenses,*[76] the BSD Licenses[77] 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 binaries.[78] Additionally, they protect the contributors and/or distributors against warranty claims of the software users, because these licenses contain a 'No Warranty Clause.'[79] 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":[80] Therefore it is incorrect to distribute BSD licensed code under another license—regardless of whether it closes the sources or not.[81]

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-closing—neither in unmodified nor in modified form—because they allow to redistribute only the binaries without also supplying the source code.[82] Finally, the BSD licenses do not protect the on-top developments against privatization.

[76] cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.

[77] 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

[78] cf. *Open Source Initiative*: The Open Source Definition, 2012, wp. §1ff.

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

[80] cf. id., ibid.

[81] 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 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. id., ibid.)

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

2.4 The protecting power of the CDDL [tbd]

As an approved *open source license,*[83] the Common Develop and Distribution 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 binaries[84]

[...]

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

As an approved *open source license,*[85] the Eclipse Public License[86] 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 binaries[87]. Furthermore, based on its patent clause,[88] the EPL protects the users also against patent disputes.[89] Besides this patent clause, the EPL contains the sections "no warranty" and "disclaimer of liability."[90] 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 avalaible' "must" incorporate a "copy" of the EPL into "each copy of the [distributed] program" or program package.[91] 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

83) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.
84) cf. *Open Source Initiative*: Common Development and Distribution License (CDDL-1.0); 2004 [n.y. of the html page itself] ⟨URL: http://opensource.org/licenses/CDDL-1.0⟩ – reference download: 2013-04-19, wp. §?.
85) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.
86) 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).
87) cf. id., l.c., wp §2a.
88) → OSLiC pp. 56
89) cf. id., l.c., wp §2b & §2c.
90) cf. id., l.c., wp §5 & §6.
91) cf. id., l.c., wp §3.

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."[92] 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 effect:[93] 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 code.[94] 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."[95] 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 to 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—for example – the GPL.

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

93) 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"

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

95) cf. id., l.c., wp §1.

2.6 The protecting power of the European Union Public License (EUPL)

As an approved *open source license,*[96] the European Union Public License[97] 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 binaries.[98] Furthermore, based on its patent clause[99], the EUPL protects the users against patent disputes.[100] Besides this patent clause, the EUPL additionally contains a "Disclaimer of Warranty" and a "Disclaimer of Liability."[101] These three elements together protect the contributors / distributors against patents disputes and warranty claims. Finally, the EUPL also protects the distributed sources against a re-closing / re-privatization 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."[102] Second, the EUPL 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 [...]"[103] This condition seems to be so important for the EUPL that the license repeats its message: in another paragraph the EUPL 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 distribute [...] the Work."[104] Based on the meaning of "Work" which is defined by the EUPL as "the Original Work and/or

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

97) 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*: EUPL-1.1/EN, 2007, wp). This EUPL has officially been translated into many languages, among others into German (cf. *Europäische Gemeinschaft a. European commission Joinup*: EUPL-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 lading page of the European host "Joinup" (cf. *Open Source Initiative*: European Union Public License, version 1.1 (EUPL-1.1; 2007 [n.y. of the html page itself] ⟨URL: `http://opensource.org/licenses/EUPL-1.1`⟩ – reference download: 2013-03-04, wp).

98) cf. id., l.c., wp §2.

99) → OSLiC pp. 57

100) cf. id., l.c., wp §2, at its end.

101) cf. id., l.c., wp §7 & §8.

102) cf. id., l.c., wp §5.

103) cf. id., ibid.

104) cf. id., l.c., wp §3.

its Derivative Works"[105] 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-)privatization, 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"[106]—while the term "Compatible Licence" is explicitly defined by a list of compatible licenses, for example the Eclipse Public License.[107]. 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 License[108] 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 license[109]. Hence the protection of the EUPL-1.1 is not as comprehensive as one might assume on the basis of the license text itself,[110] it can at most be a weak copyleft license—even if the reader might get the impression that the authors of the EUPL wished to write a strong copyleft license. Howsoever, the EUPL license does not protect the on-top developments against a privatization.

105) cf. *Open Source Initiative*: EUPL-1.1 (OSI), 2007, wp §1.

106) cf. id., l.c., wp §5.

107) cf. id., l.c., wp Appendix.

108) 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.

109) → OSLiC, p. 31

110) This kind of specifiying the protective power of the EUPL is initially presented by the FSF (cf. *Free Software Foundation*: Various Licenses and Comments about Them, 2013, 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 undoubtely true. But unfortunately, this argument does not close the hole in the protecting shield put up by the EUPL.

2.7 The protecting power of the GNU General Public License (GPL)

The GNU General Public License—also known as GPL—is maintained and offered by the Free Software Foundation and hosted as part of the well known "GNU operating system homepage."[111] Currently, there are two versions of the GPL which are classified as OSI approved open source licenses[112], the GPL-2.0[113] and the GPL-3.0. [114] Although both versions of the GPL aim for the same results and the same spirit, they differ with respect to textual and arguing structure. Therefore, it is helpful to treat these two licenses separately.

2.7.1 GPL-2.0

The protecting power of the GPL-2.0 can easily be determined: First, the license allows the users of a received software to "copy and distribute" unmodified "copies of the [...] source code"[115] as well as to "[...] modify [...] copies [...] or any portion of it, [...] and (to) distribute such modifications [...]"[116]—not only in the form of source code, but also in the form of binaries.[117] Thus—and in accordance of being an approved *open source license*[118]—the GPL-2.0 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 binaries. Second, it protects the contributors against warranty claims[119] and—based on its copyleft effect[120]—also against the loss of feedback. Third, the GPL-2.0 protects the source code itself in a nearly complete mode against privatization: even if one initially distributes only the binary version of a modification which one has generated (as a "work based on the" original) by "copying" any portion of the original work into this new derivative work,[121] then one has nevertheless to offer a possibility to get the

111) cf. *Free Software Foundation*: GNU Operating System[:] Licenses; 2011 ⟨URL: http://www.gnu.org/licenses/⟩ – reference download: 2013-03-25, wp.
112) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.
113) For the original version, offered by the FSF cf. *Free Software Foundation*: The GPL-2.0 License (FSF), 1991, wp. For the version, offered by the OSI cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp.
114) For the original version, offered by the FSF cf. *Free Software Foundation*: GNU General Public License [version 3]; 2007 [n.y. of the html page itself] ⟨URL: http://www.gnu.org/licenses/gpl.html⟩ – reference download: 2013-03-06, wp. For the version, offered by the OSI cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp.
115) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp §1.
116) cf. id., l.c., wp §2.
117) cf. id., l.c., wp §3.
118) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.
119) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp §11, §12.
120) cf. id., l.c., wp §3.
121) cf. id., l.c., wp §2.

source code[122]—namely for "the modified work as whole."[123] This modified "work based on the [original] Program" has to be read in a very broad sense; it "[...] means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language."[124] Hence, in the context of software distribution, the GPL-2.0 does not only protect the software against re-privatization, but also possible on-top developments against privatization.

But the GPL-2.0 does not protect against patent disputes[125]—neither the users, nor the contributors or distributors—and it does not protect the (modified) software which is not distributed against (re-)privatization.[126]

2.7.2 GPL-3.0

An important modification of the GPL-3.0 is evoked by the use of the new wording to "propagate" or to "convey" a "covered work": On the one hand a "covered work" denotes "either the unmodified Program or a work based on the Program". This "work based on the Program" is defined as a "modified version" of an "earlier" instance of the program which has been derived from this earlier instance by "(copying it) from or (adapting) all or part of it" in way other than exactly copying the earlier instance.[127] On the other hand, "to propagate a work" denotes "copying, distribution (with or without modification), making available to the public" and any other kind of treating the work "[...] except executing it on a computer or modifying a private copy."[128] Third, the GPL 3.0 specifies that to "convey" a work "[...] means any kind of propagation that enables other parties to make or receive copies."[129] This specification shall later on help to clarify that it is an act of distribution if the recipient himself actively copies or fetches a

122) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp §4.
123) cf. id., l.c., wp §3.
124) cf. id., l.c., wp §0.
125) → OSLiC, p. 58
126) This is a 'lack' in the GPL which the AGPL wants to close: you are indeed allowed to modify and install a GPL-2.0 licensed server software on your own machine for offering a service based on this modified software without being obliged to give your improvements back to the community. But—at least in Germany—this viewpoint seems to have to respect rigorous limits. Sometimes, it is said that even distributing software over the parts of a holding is already a distribution which—in the case of GPL-2.0 licensed software—would evoke the obligation to distribute the source code, too. [IMPORTANT: citation still needed!]
127) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp §0.
128) cf. id., l.c., wp. §0. The GPL 3.0 wants to cover the copyright systems of all countries of the world without dealing with their particular constraints directly. Therefore it generally states, that the meaning of the phrase "to propagate a work"—in the spirit of the FSF—is whatever the specific copyright system wants to be covered by these words, "[...] except executing it on a computer or modifying a private copy".
129) cf. id., l.c., wp §0.

program.

Referring to this new wording, the GPL-3.0 allows as a "basic permission" to "[...] make, run and propagate covered works [...] without conditions so long as your license otherwise remains in force."[130] This might be read as *anything is allowed without any restrictions—provided there does not exist any rule which must be respected*. Based on these specifications, the use and the modification of a GPL-3.0 program only for yourself is not restricted.[131]

So, in general—like all the other open source licenses and in accordance to the OSD[132]—also the GPL 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 binaries.[133] Furthermore, based on its patent clauses, the GPL-3.0 protects the users and the contributors of a software against patent disputes.[134] Additionally, the GPL-3.0 tries to protect the contributors or distributors against warranty claims by its well known "Disclaimer of Warranty"[135] and "Limitation of Liability"[136] which must explicitly made been known at least in each case of source code distribution.[137] Finally, the most forceful protection of the GPL-3.0 concerns the protection against the loss of feedback and against the privatization: Whenever you distribute a GPL-3.0 licensed program in the form of binaries, you have to make the source accessible, too.[138] Moreover, this obligation concerns every covered work, hence not only the unmodified original, but also any modification or adaption derived by any other kind of copying parts of the original into the "resulting work":[139] "You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License."[140] So, no doubt: the GPL wants also the source

130) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp §2.

131) In general, you have to infer that you do not have to fulfill any tasks if you are using a piece of open source software only for yourself—namely based of the fact that the particular license rules focus only on the distribution of the software, not on the private use. But in the GPL-3.0, this assertion concerning the private use becomes more explicit: It is one of your "basic permissions" to "[...] make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force". And "to propagate a work" refers to anything "[...] except executing it on a computer or modifying a private copy" (cf. id., l.c., wp. §2 and §0). Thus, the GPL-3.0 supports your total freedom on your own machine: Do whatever you want to do; anything goes—as long as you do not hand the result over to any third party in any sense.

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

133) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp §3, §4, §5, and §6.

134) → OSLiC, p. 58

135) cf. id., l.c., wp §15.

136) cf. id., l.c., wp §16.

137) cf. id., l.c., wp §4.

138) cf. id., l.c., wp §6.

139) cf. id., l.c., wp §0.

140) cf. id., l.c., wp §6.

code of all on-top developments to be published, not only the modified programs and libraries used as base of these on-top developments. The single mode of use, the GPL does not protect against privatization, is the mode of using the software only for yourself.[141]

2.8 The protecting power of the GNU Lesser General Public License (LGPL)

The LGPL is maintained and offered by the Free Software Foundation and hosted as part of the well known "GNU operating system homepage."[142] The meaning of the name *LGPL* was changed in the course of time. First, in 1991, it should be resolved as "GNU Library General Public License" and should denote the "first released version of the library GPL" which was "[...] numbered 2 because it goes with version 2 of the ordinary GPL." Today, this license is marked as "superseded by the GNU Lesser General Public License"[143]. This newer *LGPL* version from 1999 was released as "the successor of the GNU Library Public License, version 2, hence [as] the version number 2.1."[144] Finally, in June 2007, the—for now—last version of the *LGPL* was released—namely with a new structure: While GPL-2.0 and LGPL-2.1 are similar, but independent licenses, the LGPL-3.0 has to be read as an addendum to GPL-3.0. At the beginning of the LGPL-3.0 license, the content of the corresponding GPL-3.0 was included into the LGPL by the sentence that "this version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below."[145] Based on these differences, it seems to be suitable to treat the different LGPLs separately.

---

141) Quite the contrary: The GPL-3.0 explicitly allows to delegate the modification to third parties and allows to distribute the source code as working base "[...] to others for the sole purpose of having them make modifications exclusively for you [...]" (cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. §2).

142) cf. *Free Software Foundation*: The GNU OS Licenses, 2011, wp.

143) cf. *Free Software Foundation*: GNU Library General Public License [version 2.0]; 1991 [n.y. of the html page itself] ⟨URL: http://www.gnu.org/licenses/old-licenses/lgpl-2.0.html⟩ – reference download: 2013-03-25, wp.

144) cf. *Free Software Foundation*: GNU Lesser General Public License [Version 2.1]; 1999 [n.y. of the html page itself] ⟨URL: http://www.gnu.org/licenses/lgpl-2.1.html⟩ – reference download: 2013-03-06, wp.

145) cf. *Free Software Foundation*: GNU Lesser General Public License [version 3]; 2007 [n.y. of the html page itself] ⟨URL: http://www.gnu.org/copyleft/lesser.html⟩ – reference download: 2013-03-06, wp.

2.8.1 LGPL-2.1

Like the other versions of the GPL or LGPL, the LGPL-2.1 also explicitly describes its purpose as the task to "protect" the "rights" of the software users: it states that generally all "[...] the GNU General Public Licenses are intended to guarantee your freedom to share and change free software [...]"[146] Of course, the LGPL-2.1 is an approved *open source license*[147] which 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 binaries.[148] But the LGPL-2.1 does not offer any sentences to infer that it grants any patent rights to the software user.[149] So, it does not protect anyone against patent disputes, neither the users, nor the contributors / distributors. Instead of this, the LGPL-2.1 contains a special section "No Warranty" offering two paragraphs which together establish the protection of the contributors and distributors against warranty claims.[150] Finally, the LGPL-2.1 also protects the distributed sources against a re-closing / re-privatization and the contributors against the loss of feedback. For that purpose, the LGPL-2.1 on the one hand states that the recipient "[...] may modify (his) copy or copies of the Library or any portion of it [...] and copy and distribute such modifications [...]" provided that the results of these modifications are "[...] licensed at no charge to all third parties under the terms of (the LGPL-2.1)."[151] On the other hand, this LGPL version allows to distribute such modifications "in object code or executable form" provided that one accompanies these entities "[...] with the complete corresponding machine-readable source code" which itself must be distributed under the terms of the LGPL-2.1.[152]

But contrary to the GPL, the LGPL does not require to publish the code of an overarching program or any on-top development: It distinguishes the "work that *uses* the Library" from the "work *based on* the Library": First, it defines the "Library" as any "software library or work" licensed under the LGPL-2.1 and adds that "a 'work *based on* the Library' means either the Library or any derivative work under copyright law."[153] Second, it defines the "work that *uses* the Library" as any "[...] program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it" whereas this "work that *uses* the Library"—taken "in isolation"—clearly "[...] is not a derivative work of the Library [...]"[154] Third—and explictily "as an exception to

146) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp Preamble.
147) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.
148) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp §1, §2, §4.
149) → OSLiC, p. 59
150) cf. id., l.c., wp §15, §16.
151) cf. id., l.c., wp §2.
152) cf. id., l.c., wp §4.
153) cf. id., l.c., wp §0, emphasis ours.
154) cf. id., l.c., wp. §5, emphasis ours. To be exact: the LGPL states also, that this work can

the Sections above"—the LGPL-2.1 allows to "[...] combine or link a 'work that uses the Library' with the Library to produce a work containing portions of the Library, and distribute that work under terms of (one's own) choice" provided one "(accompanies) the work with the complete corresponding machine-readable source code for the Library". Together, these three specifications clearly require that one must publish / distribute the source code of the library itself—regardless, whether it is modified or not, and regardless, whether one distributes the code directly or makes 'only' written offer for receiving the source code of the library separately.[155] But these specifications do not require that one also must publish / distribute the source code of the *work that uses the library* or—as the OSLiC is using to say—the *the on-top developments*.

Thus—no surprise—it has to be inferred that the LGPL does not protect the on-top developments against a privatization. And of course, that is the reason why it is called the *GNU* Lesser *General Public License*.

2.8.2 LGPL-3.0

The LGPL-3.0 wants to be read as an extension of the GPL-3.0. For that purpose, it explicitly "[...] incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by (some) additional permissions [...]"[156] Thus, the LGPL-3.0 inherits the most parts of the protecting power of the GPL-3.0—except those parts which deal with the overarching on-top development: In opposite of the GPL-3.0, the LGPL allows to embed LGPL-3.0 licensed libraries into libraries of higher complexity[157], into on-top applications[158] and into sets of reorganized library systems.[159] Moreover, the LGPL-3.0 allows to "convey" these overarching units "under terms of (one's own) choice."[160] Therefore, one is not necessarily obliged to publish the source code of these on-top developments, too[161]—but, of course, one is obliged to publish the source code of the (modified) embedded libraries themselves.

nevertheless become a derivative work under the particular circumstances of being linked to the library. But even then, the LGPL allows to treat this 'derivative work' as a work which is not a derivative work, provided one fulfills some additional conditions. With respect to this viewpoint, the hint of the LGPL that the non-derivative work becomes a derivate work by linking it, seems not to be as crucial as one might expect.

155) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp §6.

156) cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp just before §0.

157) cf. id., l.c., wp §3.

158) cf. id., l.c., wp §4.

159) cf. id., l.c., wp §5.

160) cf. id., l.c., wp §4.

161) To be exact: The LGPL-3.0 wants to assure that "combined works" can be re-combined on the base of newer versions of the embedded library. For that purpose, one has either to use "a suitable shared libary mechanism" which allows to replace the embedded library without relinking the larger unit, or one has to publish at least "the minimal corresponding source

Based on the already described protecting power of the GPL-3.0[162] and on these additional specifications of the LGPL-3.0, one can summarize the protecting power of the LGPL-3.0 this way:

First, the LGPL protects the users against the loss of the right to use, to modify and/or to distribute the received software. Additionally, it protects them against patent disputes. Second, it protects the contributors and distributors against the loss of feedback, against warranty claims and against patent disputes. Finally, it protects the distributed software itself against re-privatization.

But the LGPL-3.0 does not protect the undistributed source code and does not protect the on-top developments against privatization.

2.9 The protecting power of the MIT license

As an approved *open source license,*[163] the MIT License[164] 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 binaries.[165] Additionally, it protects the contributors and/or distributors against warranty claims of the software users, because it contains a 'No Warranty Clause.'[166] 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."[167]

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

[code]" and a set of binaries by which the user himself can relink the overarching unit on the base of a newer version ob the embedded library (cf. *Free Software Foundation*: The LGPL-3.0 License (FSF), 2007, wp. §4)

162) → OSLiC, p. 36

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

164) '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).

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

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

167) 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".

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 code.[168] Finally, the MIT license does not protect the on-top developments against a privatization.

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

As an approved *open source license,*[169] the Mozilla Public License[170] 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 binaries.[171] Furthermore, based on its split and distributed patent clause,[172] the MPL protects the users against patent disputes.[173] Besides this patent sections, the MPL additionally contains a "Disclaimer of Warranty" and a "Limitation of Liability."[174] These three elements together protect the contributors / distributors against patents disputes and warranty claims. Finally, the MPL also protects the distributed sources against a re-closing / re-privatization and the contributors against the loss of feedback: The MPL clearly says that, on the one hand, "all distribution of

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

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

170) 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 release number for indicating that with respect to its protecting power the MPL-2.0 and the MPL-1.1 can be taken as equipollent.

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

172) → OSLiC pp. 60

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

174) cf. id., l.c., wp §6 & §7.

Covered Software in Source Code Form, including any Modifications [...] must be under the terms of this License"[175] and that, on the other hand, MPL licensed software "[...] (distributed) in Executable Form [...] must also be made available in Source Code Form [...]"[176] 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 privatization: First, the MPL does not use the term *derivative work*.[177] Instead of this, the MPL denotes the "[...] (initial) Source Code Form [...] and Modifications of such Source Code Form" by the label "Covered Software"[178]—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."[179] 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 seperate file or files, that is not covered software."[180] 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."[181] 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 work,[182] 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 files.[183] Hence, the MPL is license with a weak copyleft

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

176) cf. id., l.c., wp §3.2.

177) 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).

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

179) 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).

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

181) cf. id., l.c., wp §3.3.

182) This follows from the general meaning of a *derivative work* as a benevolent software developer would read this term (→ OSLiC, pp. 67). But again: The MPL does not focus on this general aspect; it uses its own concept of a *larger work*.

183) 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 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

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

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

As an approved *open source license,*[184] 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 binaries.[185] Furthermore, based on its patent clause,[186] the MS-PL protects the users against patent disputes.[187] 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 claims.[188] 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 part of the software,"[189] and because, second, one must also incorporate "a complete copy of this license" into one's own distribution premised one distributes the source code.[190]

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, too.[191]

---

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 Findation. 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 undoubtably be an incorporation of portions and therefore would make the incorporating file becoming a modification of the MPL licensed initial work.

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

185) 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.

186) → OSLiC pp. 61

187) cf. id., l.c., wp §2.B and §3.B.

188) cf. id., l.c., wp §2B, §3B, §3D.

189) cf. id., l.c., wp §3C.

190) cf. id., l.c., wp §3D.

191) 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⟩ – 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/

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 privatization.

2.12 The protecting power of the Postgres License (PostgreSQL)

As an approved *open source license,*[192] 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 binaries.[193] Because of its *disclaimer of warranty*, the PostgreSQL also protects the contributors / distributors against warranty claims.[194] Finally, the PostgreSQL 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."[195]

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 PostgreSQL does not protect the undistributed software or the distributed binaries against re-closings—neither in unmodified nor in modified form—because the PostgreSQL allows to (re)distribute the binaries without also supplying the sources—even if the binaries rest upon sources modified by the distributor. Finally, the PostgreSQL does not protect the on-top developments against a privatization.

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.

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

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

194) cf. id., ibid.

195) cf. id., ibid.

2.13 The protecting power of the PHP License

As an approved *open source license,*[196] 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 binaries.[197] Because of its *disclaimer of warranty,* the PHP license also protects the contributors / distributors against warranty claims.[198] 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."[199]

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.

2.14 Summary

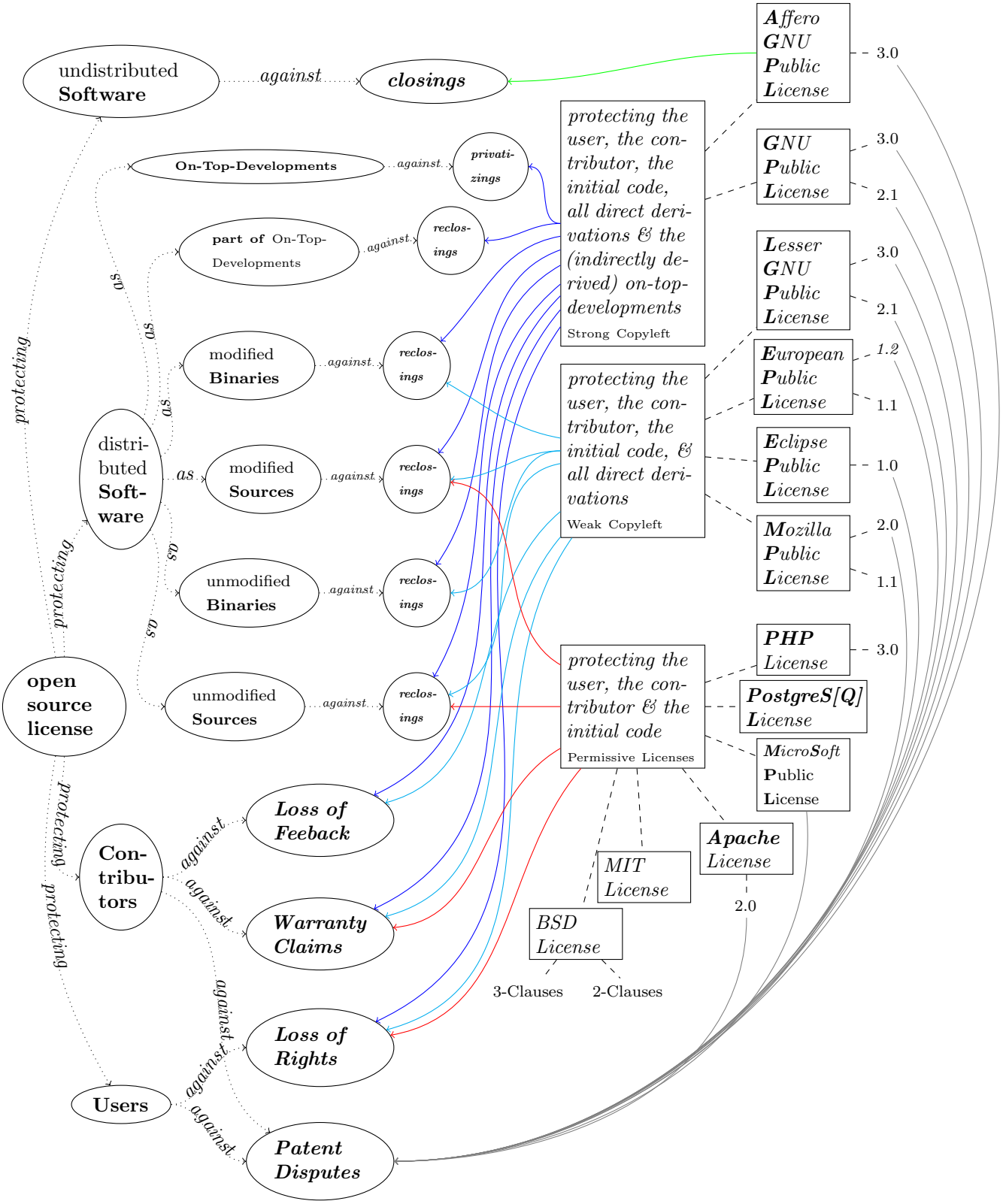
All these specifications can not only be summarized by a table,[200] but also by a mindmap as it is shown at the end of this chapter. Moreover, based on these specifications, one could generate new groups of open source licenses, new classes, like 'user protecting licenses,'[201] 'patent disputes fending licenses' up to more sophisticated taxonomies.

However, one must keep in mind that all of these grouping viewpoints do not legitimate 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 neglected.[202] So: reflecting on possible classes of open source licenses is

196) cf. *Open Source Initiative*: The Open Source Licenses, alphabetically sorted, 2012, wp.
197) 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.
198) cf. id., ibid.
199) cf. id., ibid.
200) → OSLiC, p. 28
201) all of them because all of them have to fulfill the OSD
202) 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

a good method to become familiar with the area of open source licenses. But it is not a method to determine, what needs to be done to obtain the right to use the software. For that purpose every license must be considered individually.

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.


undistributed Software
against
closings
On-Top-Developments
against
privati-zings
part of On-Top-Developments
against
reclos-ings
modified Binaries
against
reclos-ings
modified Sources
against
reclos-ings
unmodified Binaries
against
reclos-ings
unmodified Sources
against
reclos-ings
distri-buted Soft-ware
as
as
as
as
as
protecting
protecting
open source license
protecting
protecting
protecting
Con-tribu-tors
Users
against
against
against
against
against
Loss of Feeback
Warranty Claims
Loss of Rights
Patent Disputes
protecting the user, the contributor, the initial code, all direct derivations & the (indirectly derived) on-top-developments
Strong Copyleft
protecting the user, the contributor, the initial code, & all direct derivations
Weak Copyleft
protecting the user, the contributor & the initial code
Permissive Licenses
Affero GNU Public License
3.0
GNU Public License
3.0
2.1
Lesser GNU Public License
3.0
2.1
European Public License
1.2
1.1
Eclipse Public License
1.0
Mozilla Public License
2.0
1.1
PHP License
3.0
PostgreS[Q] License
MicroSoft Public License
Apache License
2.0
MIT License
BSD License
3-Clauses
2-Clauses

3 Open Source: About Some Side Effects

3.1 The problem of implicitly releasing patents

> *In this chapter, we briefly analyze the effects 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 patents.[203] One of the best known witnesses for that attitude is the GPL itself. Its preamble purports that "[...] any free program is threatened constantly by software patents."[204] 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 in patent ligitations.[205] Second, they fear a bramble of general software patents which practically prohibits them to develop open source software legally.[206] Third, they anticipate the possibility that (not quite benevolent) open source developers

---

203) For an early and elaborate 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 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)

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

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

206) cf. id., ibid.

could try to register patents with the intention of undermining the open source principles.[207]

Howsoever, regardless whether one tries to fight against software patents or not, software patents have become a reality. To abide by the law requires managing 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 clauses[208] or by several sentences distributed in the license text.[209] But why does the OSLiC have to deal with this topic, if the OSLiC does not want to participate in general discussions?

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 have to be executed for acting in accordance with the licenses. Open source patent clauses normally do not join in the game 'paying by doing.' So, actually, it does not seem to be 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 have side effects which imply that the distributors of open source software already have something done if they actually distribute 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 to and distributors 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 legally.[210] So, let us—a little simplifying and therefore only on the following few pages—name such licenses the *granting licenses*.

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

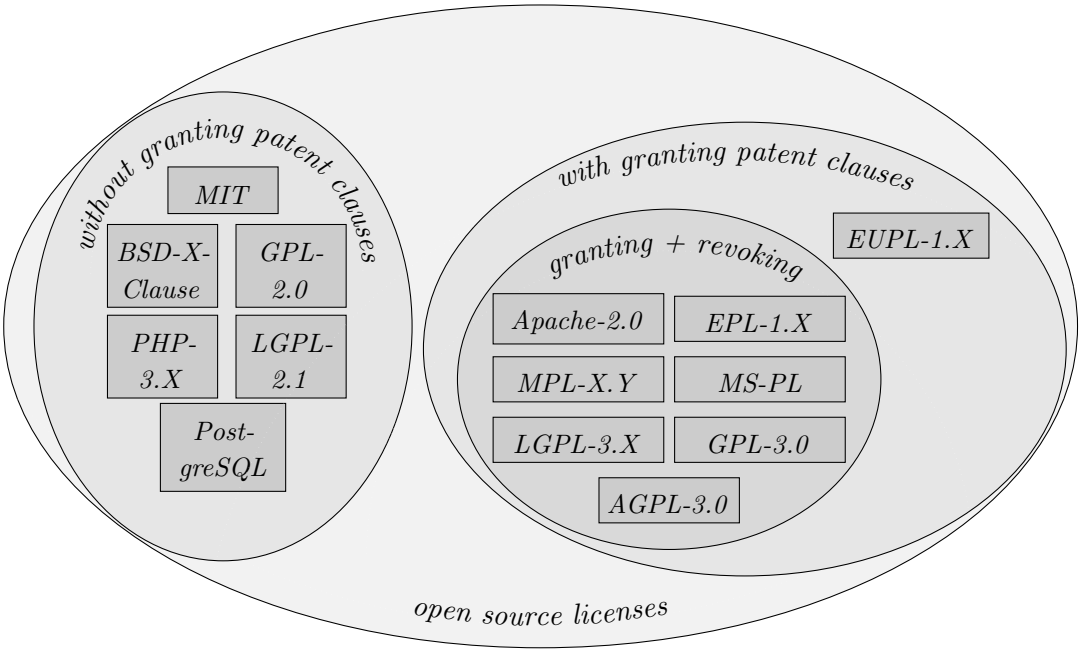
208) pars pro toto cf. *Open Source Initiative*: APL-2.0, 2004, wp §3.

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

210) There might arise a legal discussion whether even a distributor who does not contribute to the software development has to grant the necessary rights of his patent portfolio. The OSLiC does not want to participate in this discussion. We take a simple and pragmatic position: to be 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.

- 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 rights granted earlier. 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 uses 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—not even if he *tries* to release them by an open source patent clause.[211] Implictly touched patents of third parties not having contributed to the development and/or participated in the distribution can never be implicitly

[211] The EPL is one of the licenses which insists on this aspect: It the second half of its patent clause, the EPL 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*: EPL-1.0, 2005, wp §2c).

and automatically released on the base of such an (open source) patent clause: no rights, no right to release.[212] Hence: even for those open source licenses which try to protect the users, finally the users themselves must nevertheless ensure that they do not violate the patents of third parties being unwillingly touched by the way the code works or the processes in which the software is used.[213]

- In the context of a granting license, one has also to consider that contributing to and distributing 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 clause (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 force you to grant all the rights which are necessary to use the software legally. Open source contributors and distributors should know that.[214]

[212] 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.

[213] 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. Moreover, even commercial licensors can willingly or unwillingly violate patents of other persons.

[214] 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 does not wish to participate in. From the viewpoint of an open source user who only wants to have one

- 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—this second case of patent revocation also 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, distribute or 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 AGPL statements concerning patents

(prelimiary text)

The AGPL-3.0 is a license derived from the GPL-3.0: apart from the preamble and the paragraphs §11 and §13, they contain nearly the same text.[215] In §13, the AGPL explictly refers to the focus on a "remote network interaction" which shall also be able to trigger the delivery of the corresponding source code; and in §11, the AGPL establishes its specific patent clause cf. *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, §11 and §13.

Like the GPL-3.0, the AGPL-3.0 tries to protect all licensees against patent claims. This kind of protection is then established by three steps:

First, the AGPL-3.0 assures that "each contributor grants a non exclusive, world-wide, royalty free patent license under the contributor's essential patent claims, to

reliable and secure way to use open source software compliantly, one should perhaps assume that there is no difference.

215) compare *Open Source Initiative*: GNU Affero General Public License, Version 3 (AGPL-3.0); 2007 [n.y. of the html page itself] ⟨URL: http://opensource.org/licenses/AGPL-3.0⟩ – reference download: 2013-04-05, and *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, in both §1 . . . §11

make, use, sell offer for sale, import and otherwise run, modify and propagate the contents of its contributor version."[216] Furthermore, the patent license defines that this patent license granted by the contributor is automatically extended to all downstream 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 work based on the program.[217]

Second, the AGPL enforces not only the grant of patent licenses by the "contributors," the license even requires the same from licensees who distributes the program unchanged: "If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it."[218]

Finally, the AGPL-3.0 introduces an revoking clause by stating that a licensee "[...] may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it"[219] and that this licensee "automatically" loses the rights granted by the AGPL-3.0 "including any patent licenses" if he tries to propagate or modify a covered work against the regulations of the AGPL-3.0.[220]

According to that, the AGPL-3.0 is like the GPL-3.0 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 installs any litigation regarding an infringement of patents, then the rights granted to him are revoked.

3.1.2 Apache-2.0 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 sentences.[221] 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 [...]"[222]

---

216) cf. *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, wp §11.
217) cf. id., ibid.
218) cf. id., ibid.
219) cf. id., l.c., wp §10.
220) cf. id., l.c., wp §8.
221) cf. *Open Source Initiative*: APL-2.0, 2004, wp §3.
222) cf. id., l.c., wp §4.3.

The one core message of the Apache-2.0 patent clause is 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 [...]"[223]

The second core message of the Apache-2.0 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 [...]"[224]

The third message of the Apache-2.0 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".[225]

Thus, the Apache-2.0 is—as we use 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 the infringement of patents, then the rights granted to you are revoked.

3.1.3 CDDL statements concerning patents

The patent clauses of the CDDL are similiar in spirit to the Apache License: The license grants rights to each contributors patents that are neccessarily infringed by distributing or using the software. The license also revokes all rights granted to someone who files a patent litigation with respect to the software against any contributor. The CDDL differs from other licenses in that the litigant does not lose his rights automatically and immediately but gets a grace period of 60 days. If he withdraws his claims during this period, the license granted to him will not be terminated.

The actual wording used in the CDDL is complicated by the fact that the CDDL distinguished between the "Initial Developer" and other "Contributors." A "Contributor" receives a version of the software to which he then adds some "Modifications" thus creating the "Contributor Version." For all practical purposes we can treat the "Initial Developer" as another contributor who happens to not receive any software and whose "Contributor Version" (officially called "Original Software") equals his "Modifications."

---

223) cf. *Open Source Initiative*: APL-2.0, 2004, wp §3. The "Contributor," "Work," and "You" are defined in §1: *Contributor* refers 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* denotes the licensees.

224) cf. id., ibid.

225) cf. id., ibid.

The patent licenses are granted in the clause (b) of the sections titled "The Initial Developer Grant"[226] and "Contributor Grant."[227] Each contributor grants the licensee "a world-wide, royalty-free, non-exclusive license under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version [...], to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor [...]; and (2) the combination of Modifications made by that Contributor with its Contributor Version [...]" This limits the patent license to patents infringed by code present in the contributor version. And clause (d) limits the grant even further to exclude "infringements caused by[...]third party modifications of Contributor Version"[228] or Covered Software in the absence of Modifications made by that Contributor.[229] This ensures that no contributor is required to tolerate an infringement of his patents caused by code modified after he made his contribution and, in particular, it is not possible to remove the contributors modifications completely without also removing all other causes of infringement of the patent claims because the patent license does not carry over to such a use of the software.

The section titled "TERMINATION" contains the usual defense against patent infringement claims by declaring that any such claim against a "Participant[230] [...] alleging that the Participant Software [...] directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You[231] [...] under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless [...] You withdraw Your claim [...] against such Participant either unilaterally or pursuant to a written agreement with Participant."

Thus, not only has the Participant to actively initiate the termination of the licenses, the licensee also has 60 days to either settle the case by an agreement with the Participant or to withdraw his claims.

3.1.4 EPL statements concerning patents

The Eclipse Public License treats the patents 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

226) cf. *Open Source Initiative*: The CDDL-1.0, 2004, wp §2.1(b).
227) cf. id., l.c., wp §2.2(b).
228) cf. id., l.c., wp §2.2(d).
229) cf. id., ibid.
230) The "Contributor" or "Initial Developer" against whom the claim is made
231) The party making the patent infringement claim

the Contribution of such Contributor, if any, in source code and object code form."[232] 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 combination.[233] Finally, the EPL hints to the general fact that 3^{rd} party patents not belonging to the contributors can never be implicity be released by such a patent clause. Moreover, it gives 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."[234]

Like other open source licenses, 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) Recipient's rights [...] shall terminate [...]"[235]

Thus, the EPL, too, is a granting and a revoking license: At first you are granted the use of 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.

3.1.5 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."[236] 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"[237]. Thus, the EUPL is—as we are using to say in this chapter—a granting license but not a revoking license.

3.1.6 GPL statements concerning patents

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

232) cf. *Open Source Initiative*: EPL-1.0, 2005, wp §2.b.
233) cf. id., ibid.
234) cf. id., l.c., wp §2.c.
235) cf. id., l.c., wp §7.
236) cf. *Open Source Initiative*: EUPL-1.1 (OSI), 2007, wp end of §2.
237) cf. id., l.c., wp §12.

3.1.6.1 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"[238]. Unfortunately, this specification is only an indirect claim which needs a lot of arguing for establishing a protective 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 constraints.[239] And finally the GPL-2.0 allows the original copyright holder to "add an explicit geographical distribution limitation excluding [...] countries" provided that these countries "[...] (restict) the distribution and/or use of the library [...] by patents [...]"[240] 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 neither a granting nor a revoking license.

3.1.6.2 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."[241]. 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."[242] Second, the GPL-3.0 defines that this patent license granted by the

238) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp Preamble.
239) cf. id., l.c., wp §11.
240) cf. id., l.c., wp §12.
241) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp Preamble.
242) cf. id., l.c., wp §11.

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."[243] 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."[244] And finally, the GPL-3.0 also introduces a 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"[245] 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-3.0.[246]

Thus, GPL-3.0 is 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 you—with respect to the software—install any litigation concerning an infringement of patents, then the rights granted to you are revoked.

3.1.7 LGPL statements concerning patents

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

3.1.7.1 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."[247] Unfortunately, this specification is again only an

243) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp §11.
244) cf. id., ibid.
245) cf. id., l.c., wp §10.
246) cf. id., l.c., wp §8.
247) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp Preamble.

indirect claim which needs a lot of arguing to establish a protective effect against patent disputes. Howsoever, this paragraph of the LGPL-2.1 does not directly grant any rights to the software users to use necessary patents.

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 conditions of this license" so that a waiving of the use of the software is the only way to fulfill both constraints.[248] And finally the LGPL-2.1 allows the original copyright holder to "add an explicit geographical distribution limitation excluding [...] countries" provided that these countries "[...] (restict) the distribution and/or use of the library [...] by patents [...]"[249] Based on these statements, one cannot infer that the LGPL grants any patent rights to the software user, neither directly, nor indirectly.

Thus, the LGPL-2.1 is neither a granting nor revoking license.

3.1.7.2 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."[250] The LGPL-3.0 itself does not contain the word 'patent,' but the GPL-3.0 does.[251] So, the LGPL-3.0 inherits its patent clause from the GPL-3.0 which is—as we already described[252]—a granting and a revoking license.

3.1.8 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"[253] Second, it hihlights some "limitations."[254] And finally, the MPL introduces a revoking clause which signifies that the rights, granted to the licensee "[...] by

248) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp §11.
249) cf. id., l.c., wp §12.
250) cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp wp.
251) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp §11.
252) → OSLiC, p. 58
253) cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp §2.1, esp. §2.1.b.
254) cf. id., l.c., wp §2.3.

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 [...]"[255]

Thus, the MPL is a granting license and a revoking license.

3.1.9 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."[256] 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."[257]

Thus, the MS-PL is 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 install any litigation concerning an infringement of patents with respect to the software, then the rights granted to you are revoked.

3.2 Excursion: Why linking is a secondary criterium

> *Distributing statically or dynamically linked software is often discussed as a problem (and sometimes as a solution) for acting compliantly. In this chapter, we briefly discuss why this aspect can mostly be ignored and why it does not help to determine the existence of a derivative work.*

In some earlier versions of the OSLiC, its finder subclassified some use cases with respect to the way an application was 'composed' as a larger unit: In the previous form for gathering the necessary information, the OSLiC user had to answer whether he *was going to combine the received open source software with other software components by linking them together statically, by linking them dynamically, or by textually including (parts of) the open source software into a larger unit.* Today, this question has totally been erased. The authors could convince themselves that it is not necessary to consider this aspect.

Of course, we know that being linked statically or dynamically is often and deeply discussed by license experts.[258] It seems to be an important aspect:

[255] cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp §5.2.
[256] cf. *Open Source Initiative*: MS-PL, 2013, wp §2.B.
[257] cf. id., l.c., wp §3.B.
[258] Even on the *European Legal and Licensing Workshop, 2013* in Amsterdam, there was given an excellent lecture concerning the nature and concequences of linking elf files.

[TBD: Discussion of the literature]

So, let us start with some undeniable facts: The OSLiC deals with the Apache-2.0 license,[259] the BSD 2-Clause license,[260] the BSD 3-Clause license,[261] the MIT license,[262] the MS-PL,[263] the PostgreSQL,[264] and the PHP license[265] as instances of permissive licenses. Additionally, the OSLiC treats the EPL,[266] the EUPL,[267] the LGPL,[268] and the MPL[269] as licenses with weak copyleft. Finally, the OSLiC thoroughly discusses the GPL[270] and the AGPL[271] as licenses with strong copyleft.[272]

Only three of these licenses mention the word *linking* (or variants of it): Using the command `grep -i link * | grep -v "<link\|links\|skip-link"` in a shell—executed as an operation on a set of html formatted license files—directly shows that only the AGPL-3.0, the Apache-2.0, the GPL-2.0, the GPL-3.0, the LGPL-2.1 and the LGPL-3.0 are using mutations of the word *linking*. Additionally, the results of the command `grep -i statical *` show that only the LGPL-2.1 uses the word 'statical,' while using the command `grep -i dynamical *` only hints to the AGPL-3.0 and the GPL-3.0. Finally, the command `grep -i "shared" *`—executed on the same set of files—shows that the term *shared libary* is also only used by these licenses.

This analysis already indicates that being statically or dynamically linked might not be as important for acting compliantly as it is often suggested. If one reads the concrete statements, then one can see, that acting compliantly depends only slightly and only rarely on the kind of being 'combined':

Apache-2.0: This version of the Apache license uses the word *link* only once for stating that "[...] Derivative Works shall not include works that remain separable from, or merely link [...] to the interfaces of, the Work and

259) cf. *Open Source Initiative*: APL-2.0, 2004, wp.
260) cf. *Open Source Initiative*: The BSD 2-Clause License, 2012, wp.
261) cf. *Open Source Initiative*: The BSD 3-Clause License, 2012, wp.
262) cf. *Open Source Initiative*: The MIT License, 2012, wp.
263) cf. *Open Source Initiative*: MS-PL, 2013, wp.
264) cf. *Open Source Initiative*: PostgreSQL License, 2013, wp.
265) cf. *Open Source Initiative*: PHP-3.0, 2013, wp.
266) cf. *Open Source Initiative*: EPL-1.0, 2005, wp.
267) cf. *Open Source Initiative*: EUPL-1.1 (OSI), 2007, wp.
268) For LGPL-2.1 see cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp. For LGPL-3.0 see cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp.
269) cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp.
270) For GPL-2.0 see cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp For GPL-3.0 see cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp
271) cf. *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, wp.
272) You can find html based instances of these licenses in the OSLiC directory 'licenses.' They have been downloaded from the OSI pages. All of the following statements refer to these files.

Derivative Works thereof."[273] Thus, the Apache-2.0 does not use the criteria *being linked* for determining a derivative work, neither *being linked* in general, nor *being statically linked*, nor being *dynamically linked*. Hence, for acting in accordance to the Apache-2.0, this class of attributes can completely be ignored.

GPL-3.0: The GPL-3.0 uses the word *link* three times: First, it defines the "'Corresponding Source' for a work in object code form [...as] all the source code needed to generate, install, and [...] run the object code and to modify the work [...]". Additionally the GPL-3.0 also explains in this context that this definition shall include "[...] the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require"[274]. Second, the GPL-3.0 allows "[...] to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work."[275] Finally, the GPL-3.0 explains that "the GNU General Public License [itself] does not permit incorporating your program into proprietary programs" and that the LGPL might be a better license for those licensors who have written a "subroutine library [...] and may consider it more useful to permit linking proprietary applications with the library [...]"[276].

So, also in this text, the features *statically linked* or *dynamically linked* are not used to trigger any license fulfilling actions. The conditions for "Conveying Modified [...] Versions" refer to the "work based on the Program"[277] which itself denotes a "'modified version' of the earlier work"[278]. Moreover, the licensee—as modifier, distributor, and subsequent licensor—is required by the GPL-3.0 "[...] to license the entire work [which has been developed on the base of a GPL-3.0 component], as a whole, under this License to anyone who comes into possession of a copy"[279]. The GPL-3.0 does not limit this claim—especially not by referring to a mode of being linked. Hence, also with respect to the GPL-3.0, one can completely ignore these features of the software, its use and its distribution for determining how to use the software compliantly.

AGPL-3.0: Concerning the use and the meaning of the words *dynamically* and *linking*, the AGPL-3.0 exactly follows the structure of the GPL-3.0: first the terms arise in the context of defining the "Corresponding Source";[280] then

273) cf. *Open Source Initiative*: APL-2.0, 2004, wp. §0.
274) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. §0.
275) cf. id., l.c., wp. §13.
276) cf. id., l.c., wp. last parapgraph.
277) cf. id., l.c., wp. §5.
278) cf. id., l.c., wp. §0.
279) cf. id., l.c., wp. §5.
280) cf. *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, wp. §0.

the word *link* helps to say that AGPL and GPL are compatible licenses;[281] and finally the word *link* is used to hint to the LGPL.[282] So, again, one can ignore the feature of being statically or dynamically linked if one wants to determine how to use the software compliantly.

GPL-2.0: In the GPL-2.0, the word *link* only arises in the context of hinting to the LGPL.[283] Moreover, the words *statical* and *dynamical* are not used in this text—not at all and in no sense: the copy left feature of the GPL depends 'only' on a specification which refers to a "work based on the Program [...] that in whole or in part contains or is derived from the Program or any part thereof [...]"[284] Thus, even in this old version of the GPL, the criteria of being linked—in which way ever—does not trigger any task for using the software compliantly.

LGPL-3.0: In this license, variants of the word *link* are used to define the concept of a "Combined Work" which shall be the name for a "[...] work produced by combining or linking an Application with the Library."[285] In the end the LGPL-3.0 allows to "[...] convey a Combined Work under terms of your choice [...]", provided that one distributes also all material (including the object files of the overarching on-top developments) necessary for enabling the receiver to relink the whole product with a later version of the library or that one presupposes the use of a "suitable shared library mechanism" so that the receiver can update the library simply by replacing the binary library file[286]. For fulfilling these conditions it is sufficient to require that a distributor shall *either distribute the on-top development and the library in the form of dynamically linkable parts or distribute the statically linked application together with a written offer, valid for at least three years, to give the user all object-files of the on-top development and the library, so that he can relink the application on its own behalf.*

LGPL-2.1: Even if the LGPL-2.1 argues more sophistically than all the other licenses, in its preamble this license clearly states what it wants to evoke: "If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. [...]"[287] For that purpose, the LGPL-2.1 defines at the beginning that if "a program is linked with a library, whether statically or using a shared library, [then] the combination of the two is legally speaking a combined work, a derivative of the original

281) cf. *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, wp. §13.
282) cf. id., l.c., wp. §5.
283) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp. last paragraph.
284) cf. id., l.c., wp. §2.
285) cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp. §0.
286) cf. id., l.c., wp. §4.
287) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp. preamble.

library":[288] On the one hand a "work that uses the Libary"—which is only "[...] designed to work with the Library by being compiled or linked with it [...]"—"[...] in isolation, is not a derivative work of the library [...]". On the other hand, it is no question for the LGPL-2.1, that "linking a 'work that uses the Library' with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library)."[289] But then—"as an exeption"—the LGPL-2.1 allows to "[...] combine or link a 'work that uses the Library' with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice". The right to do this is granted provided that the distributor either presupposes the use of a "suitable shared library mechanism" or that he distributes also the complete material (including the object files of the overarching on-top developments) which is necessary to enable the receiver to relink the whole product with a later incoming newer version of the library[290]. Again, for fulfilling all these conditions it is sufficient to require that a distributor shall *either distribute the on-top development and the library in the form of dynamically linkable parts or distribute the statically linked application together with a written offer, valid for at least three years, to give the user all object-files of the on-top development and the library, so that he can relink the application on its own behalf.*

Thus, with respect to this analysis, we can conclude that—in general—there is no need to investigate whether one wants to distribute software in the form of statically or dynamically linked binaries for deriving the necessary tasks to distribute this software compliantly. Instead of this, we can directly incorporate those doings into the task lists of the LGPL what has been discovered as sufficient doings. Moreover, it is also sufficient to insert this statement only in the task list of the LGPL. There is no need to generalize this discussion. So, we could simplify our form offered to gather the information to find the adequate license fulfilling task list.

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

This chapter briefly discusses aspects of being a derivated pieces of software which have to be known for using open source software compliantly. As usually, the OSLiC only tries to find one safe interpretation. The authors know that there exist many other ways to consider this topic. So, if you feel, that the viewpoint of the OSLiC does not fit the specific circumstances of your particular case, do not

[288] cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp. preamble.
[289] cf. id., l.c., wp. §5.
[290] cf. id., l.c., wp. §6, §6b and §6c together with §6c.

hesitate to ask your own lawyer. But if you agree with the OSLiC, be aware that you dealing with this topic from the viewpoint of a benevolent user.

Let us outline the argumentation:

The meaning 'derivative work' must be known! Many open source licenses use the term 'derivative work,'[291] either directly or indirectly in form of the word 'modification.'[292] [Write a table as survey] And nearly all licenses that are using the term 'derivative work' etc., are linking tasks that must be executed to comply with the corresponding license, to the precondition that something is a derivative work. [table survey] **Hence, for acting in accordance with such a license, it has to be known what a derivate work is.**

Unfortunately the meaning is not clearly fixed. There exist different readings of the term 'derivative work.' [specify the differences and cite the sources] **Hence, it is not as clear what a derivative work is as one could wish**

So, let us 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 with their assistance. So, one should be able to answer the question under which circumstances a piece of software is a 'derivative work' of another piece of software based on two principles:

- Let us argue from the viewpoint 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 perform the required tasks.

We generalize a specific viewpoint of the LGPL. It uses three terms:

"library" is defined as "a collection of software functions and/or data prepared so as to be conveniently linked with application programs."[293]

"work based on the library" is defined as "either the library or any derivative work."[294]

"work that uses the library" is defined as something which initially "[...] is not a derivative work of the library [...]" but can become a derivative work by being combined / linked to the library it uses.[295]

[291] cite the sources
[292] cite the sources
[293] cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp §0.
[294] cf. id., ibid.
[295] cf. id., l.c., wp §5.

Following these specifications, one has to conclude that *derivative works* of the library can be drieved in two different ways: First, the library itself can be enhanced without changing the character of being a library. Then, of course, the resulting library is a derivative work of the initial library. Second, an overaching program can use the library by calling functions, methods, or data offered by the library. In this case, the overarching program functionally depends on the library and is a derivative work (as soon as it is linked to the library).

This viewpoint can be generalized: snippets, modules, plugins can be enhanced and used by overarching programs or even by more complex libraries. Based on this viewpoint—which should finally be formulated as the viewpoint of a benevolent impartial developer—the OSLiC uses the following 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 derivatve work of the source file.[296]

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

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

And here are some applications of these rules:

- **Enlarging an existing source file by an external text creates a derivative work!** Why? *Because you are going to reuse the external code for simplifying our life.* [see 'Copy Case']
- **Reducing a source file creates a derivative work!** Why? *Because you are going to prepare the given file(s) for a better reuse.* [see 'Modify-Case']
- **Replacing something in a source file creates a derivative work!** Why? *Because you are going to reuse parts of the existing code for simplifying your life.* [see 'Modify Case']
- **Integrating a foreign snippet into an existing source code creates a derivative work!** Why? *Because you are going to simplify your life by reusing both, the foreign snippet and the original file.* [see 'Copy Case' and 'Modify-Case']
- **Refactoring a given work by extracting a function / method into an autonomous file creates a derivative work in two respects!** Why? *Because, first, all modified / generated files depend on the original file and,*

[296] Be careful: this case must still be distinguished from the case of an automatic inclusion (header files, script libraries) during the compilation / execution: Inclusion of header files alone should not create a derivative work.

second, because those function calls in the files introduce a dependecy on the file defining the function itself. [see 'Modify-Case' and 'Call-Case]

- **Calling a function - served by a defining module - lets the calling file become a derivative work of the serving module!** Why? *Because you are going to simplify your life by reusing an already prepared work (often offered by other developers).* [see 'Call-Case']
- **By calling elements - served by a defining library - the calling file becomes a derivative work of the serving library!** Why? *Because you are going to simplify your life by reusing an already prepared work (often offered by other developers).* [see 'Call-Case'][297]

And now some additional 'ideas' which might invite to be discussed:

- **Does a plugin depend on its framework? No.** Why? *Because it is like a module: it offers a function (normally without using a function, offered by the framework itself).*
- **Does a framework depend on its plugin? Let us try to answer: Sometimes yes, sometimes no.** Why? *If the framwork crashes when it is missing its plugin, then it clearly depends on the plugin. No doubt. It is simply not autonomous. But if it does not crash, then it perfectly does for which it has been designed: it is offering a place which might be filled by the plugin, but not necessarily. This kind of a framework is like an application listing to a port for getting data which it shall process and which are served by another application.*
- **Does a program using inter process communication depend on its IO-partners? Definitely not!** Why? *Because, otherwise, we need not discuss all these cases, every thing would depend on everything—in each running system.*

[. . . TBD . . .]

3.4 Excursion: Reverse Engineering and Open Source

Beyond any doubt, the LGPL mentions "reverse engineering" literally[298] for indicating that reverse engineering in any respect must be allowed to use and

[297] In this context, you may sometimes read that one has to differentiate the defining file (for example the C-code) and the declaring file (for example the C-Header). But in our view, it is not so important to make such a difference: The using file, which includes the declaring header file depends on the defining source code file ('Call-Case'). So, one can ignore the formal dependance on the declaring header file ('Copy-Case').

[298] For the LGPL-v2 cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp., §6; for the LGPL-v3 cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp., §4

distribute LGPL software compliantly:

> *"[...] you may [...] distribute a work (containing portions of the Library) under terms of your choice, provided that the terms permit [...]* reverse engineering *[...]"* [299]

There are three strategies for dealing with such provisions: one can try to fully honor its meaning, one can mitigate its meaning, or one can avoid to discuss this requirement altogether:

A first group of well known open source experts take the sentence of the LGPL-v2 as a strict rule which requires that one has to allow reverse engineering of the whole software product if one embeds any LGPL licensed component into that product[300].

A second group of well known and knowledgeable open source experts signify that the LGPL-v2 indeed literally contains a strict rule, but that this rule actually is not meant as it sounds: For example, two of these experts explain that "these requirements on the licensed combination require that the license chosen not prohibit the customer's modification and reverse engineering for debugging these

---

299) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp, §6. The LGPL-v2 uses the capitalized word "Library" for denoting a library which "[...] has been distributed under (the) terms" of the LGPL-v2. (cf. id., l.c., wp, §0). Inside of our LGPL chapter(s) we follow this interpretation.

300) For example, a very trustworthy German expert states that the LGPL-2.1 generally requires that a distributor of a program which accesses a LGPL-2.1 licensed library, must grant his customer also the right to modify the accessing program and hence also the right to execute reverse engineering. Literally the German text says:

> "Ziffer 6 LGPLv2.1 knüpft die Erlaubnis, das zugreifende Programm unter beliebigen Lizenzbestimmungen verbreiten zu drüfen, an eine Reihe von Verpflichtungen, die in der Praxis oft übersehen werden: Zunächst muss dem Kunden, dem die Software geliefert wird, die Veränderung des zugreifenden Programms gestattet werden und zu diesem Zweck auch ein Reverse Engineering zur Fehlerbehebung. *Dies dürfte alle Formen des Debugging und das Dekompilieren des zugreifenden Programms umfassen.*" (cf. *Jaeger a. Metzger*: Open Source Software. Rechtliche Rahmenbedingungen der Freien Software, 2011, pp. 81; emphasis KR).

At first glance, also "copyleft.org" – the "[...] collaborative project to create and disseminate useful information, tutorial material, and new policy ideas regarding all forms of copyleft licensing" (cf. *copyleft.org*: What is copyleft.org; n.l, 2014 ⟨URL: http://copyleft.org/⟩ – reference download: 2014-12-15, wp.) – could be taken as another witness for such an attitude of strict reading: Some of its contributors elucidate in a chapter dealing with "special topics in compliance" that "the license of the whole work must [sic!] permit 'reverse engineering for debugging such modifications' to the library" and that one therefore " should take care that the EULA used for the Application does not contradict this permission"(cf. *Kuhn, Bradley M.* et al.: Copyleft and the GNU General Public License: A Comprehensive Tutorial and Guide; n.l, 2014 ⟨URL: http://copyleft.org/guide/comprehensive-gpl-guide.pdf⟩ – reference download: 2014-12-15, p. 86

modifications in the work as a whole". But then they directly add the limitation, that "in practice, enforcement history suggests, it means that the license terms chosen may not prohibit modification and reverse engineering for debugging of modification in the LGPL'd code included in the combination"[301].

Finally, a third group of experts prefers not to discuss the problem of reverse engineering, although this technique is literally mentioned in the license and although they want explain how to use GPL/LGPL licensed software compliantly[302].

This situation must bother companies and people who want to use open source software compliantly and who therefore are looking for guidance. Particularly it disturbs those who want to protect their business relevant software. At the end, they might consider that this sentence is not consistently understood by the open source community itself. And – as far as we know – at least some of these companies preventively prohibit their developers to embed LGPL licensed components into programs which contain business relevant techniques. Unfortunately, this consequence does not only obstruct the access to a large set of well written free software, but it is scarcely possible to obey such an interdiction consequently: The glibc, which enables the programs to talk with the kernel of the GNU/Linux system[303], is licensed under the LGPL[304]. And hence, this library is indirectly

[301] cf. *Moglen, Even* a. *Mishi Choudhary*: Software Freedom Law Center Guide to GPL Compliance, 2nd Edition; 2014 ⟨URL: https://www.softwarefreedom.org/resources/2014/SFLC-Guide_to_GPL_Compliance_2d_ed.html⟩ – reference download: 2014-12-15, wp., chapter LGPLv2.1, section 6. Such a mitigation can also be found in the tutorial of copyleft.org: After they have summarized the LGPL-v2 sentence as a strict rule, they directly continue, that one "[...] must refrain from license terms on works based on the licensed work that prohibit replacement of the licensed components of the larger non-LGPL'd work, or prohibit decompilation or reverse engineering in order to enhance or fix bugs in the LGPL'd components" (cf. *Kuhn* et al.: Copyleft and the GNU General Public License, 2014, p. 86). This added specification indicates, that one only has to facilitate the modification of the library and that reverse engineering can be ignored as long as there are other ways to improve the embedded library.

[302] An article of Terry J. Ilardi might be taken as a first witness of this third strategy: he profoundly explains the essence of the LGPL, he especially discusses §6, and he delivers applicable rules like "DO NOT statically link to LGPL [...] code if you wish to keep your program proprietary". But he does not discuss *reverse engineering* (cf. *Ilardi, Terry J.*: Common OSS License Problems; n.l, 2010 ⟨URL: http://www2.aipla.org/html/spring/2010/papers/Ilardi_Paper.pdf⟩ – reference download: 2014-12-16, pp. 5f). Similarily argues Rosen (cf. *Rosen*: Open Source Licensing, 2005, pp. 121ff). And – despite their comments on reverse engineering in the specific chapter *special topics in compliance* – the copyleft.org document can also be taken as an instance of this attitude: Although its authors recommend to "study chapter 10 carefully" for establishing an adequate "compliance with LGPLv2.1" (cf. *Kuhn* et al.: Copyleft and the GNU General Public License, 2014, p. 86), this chapter 10 – dedicated to the meaning of the "Lesser GPL" – does not deal with reverse engineering, although it discusses the §6 of the LGPLv2.1 in depth (cf. id., l.c., pp. 56ff, esp. 60f).

[303] cf. http://www.gnu.org/software/libc/

[304] cf. http://en.wikipedia.org/wiki/GNU_C_Library

linked to or combined with any program running on the GNU/Linux system. So, if the LGPL-v2 indeed required, that reverse engineering of every program must be allowed, which contains portions of any LGPL Library, then every GNU/Linux user would be allowed to examine every program running on GNU/Linux by *reverse engineering*, simply, because finally every 'GNU/Linux program' is linked to or combined with the glibc[305]. In other words: if the LGPL indeed required the permission of reverse engineering, then every program executed on GNU/Linux may be reverse engineered.

But an exhaustive reading of the LGPL-v2 strongly indicates, that there must be another valid, more 'liquid' understanding of the LGPL: The preamble explains the reason for offering another weaker license beside the GPL. It says that "[...] on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard" and that therefore it could be strategicly necessary to "allow [...] non-free programs [...] to use the library" without enforcing that these programs become free software too[306].

So, if the LGPL had indeed determined that every program linked to or combined with any LGPL library may be reverse engineered, then the LGPL would have an effect contrary to its own intention. It would have introduced something like *'security by obscurity'*: First, the LGPL would allow to protect the internals of your own work against investigation by allowing to keep the code of the non-free program using the library a scecret[307]. But then – in the end – the LGPL would also allow the user to reverse engineer the received binary and hence would enable him to nevertheless discover all internals[308]. Hence, finally the LGPL-v2 would undermine its own raison d'ètre introduced by its inventors: under such circumstances there probably would have been less hope that any LGPL library could have become a defacto standard.

We know that the inventors of the GNU licenses and GNU software are very sophisticated experts. They never would have published such an inconsistent document. So, this dissent read in(to) the document is a strong indicator for the fact, that there must be a better way to understand the license. And thus, it is up to us, the followers, to explicate a more adequate interpretation. Of course, such an interpretation must be grounded on the written text. No doubt: we, the scholars, are not allowed to add our own wishes. We must read the license very strictly. We have to deduce 'understandings' only by matching the interpretations explicitly and reasonably back to the license text itself.

---

305) This conclusion might surprise. But it is inferred with exactly the same arguments as the conclusion, that without a licence offering a weaker copyleft every program would have been licensed under the GPL. The copyleft.org document explains this argumentation in great detail (cf. *Kuhn* et al.: Copyleft and the GNU General Public License, 2014, pp. 56f).

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

307) The weak copyleft has been introduced for encouring the widest possible use of the library.

308) It would only cost a little more effort - as security by obscurity indicates.

Encouraged by the indication that a better understanding of the LGPL may exist and contrary to the other strategies, we are going to prove that there is a valid way to compliantly distribute any open source based software without permitting reverse engineering: We want to show that none of the main open source licenses[309] requires reverse engineering if the work using the open source Library is distributed in form of dynamically linkable files. In particular, we are going to prove that one even has not to allow reverse engineering of the work using an LGPL Library, if one distributes that work as dynamically linkable files. And we want to show that in all other cases one has at least to fear that one has implicitly allowed the reverse engineering of the work using the open source Library if one distribute that work. In particular, we want to prove that one has to fear this implicitly given permission even if one distributes a work using a library licensed under any permissive license[310].

In general, we hope that our analysis, grounded on the license text itself, will support companies and people to compliantly use open source software more often and with less hesitation due to the fear that they have to deliver themselves unclear textual aspects.

But – with respect to the discussion about this text in the OSI and Free Sofwtare Mailing lists – we have to add a disclaimer here: The license text alone is not all. In the concrete situation of using free and open source software, it is the intention of the licensor which has to be respected. Or in the words of Eben Moglen:

> *A license is, by definition, a unilateral permission to make use of the property or intangible rights of another. The measure of the permission*

[309] Just as the OSLiC, also this part focuses only on the most important open source licenses (cf. https://www.blackducksoftware.com/resources/data/top-20-open-source-licenses wp.): the Apache license (cf. *Open Source Initiative*: APL-2.0, 2004, wp.), the BSD licenses (cf. *Open Source Initiative*: The BSD 3-Clause License, 2012, wp. and cf. *Open Source Initiative*: The BSD 2-Clause License, 2012, wp.)), the MIT license (cf. *Open Source Initiative*: The MIT License, 2012, wp.), the MS-PL (cf. *Open Source Initiative*: MS-PL, 2013, wp.), the PostgreSXQL (cf. *Open Source Initiative*: PostgreSQL License, 2013, wp.), the PHP license (cf. *Open Source Initiative*: PHP-3.0, 2013, wp.), the EPL (cf. *Open Source Initiative*: EPL-1.0, 2005, wp.), the EUPL (cf. *Open Source Initiative*: EUPL-1.1 (OSI), 2007, wp.), the MPL (cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp.), the LGPLs (cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp. and cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp.), the GPLs (cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp. and cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp.) and the AGPL (cf. *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, wp.)

[310] By the way, our analysis should also provide proof that the LGPL is not something like a 'poisoned' license containing "an imprenetrable maze of technology babble" which "[...] should not be in a general-purpose software license" (cf. *Rosen*: Open Source Licensing, 2005, p. 124). The challenge of the today's descendants is to understand the former inventors of the GNU licenses and their way to think about computing - including all the hassle the computing language C might provoke.

is the intention of the party giving it.[311]

Nevertheless, we believe that each text firstly has its own inherent independent meaning and message. But – of course, in the specific situation of legally contending about the practical consequences of a license, one has indeed to consider what the specfic licensor really had had in his mind, when he released his work. One has to consider his intention.

So, the pure textual meaning of the license might be overloaded or overwritten by some external facts, traditions or understandings, not founded on the license text itself. The problem with this legal fact is, that in a concrete legal case, one has to prove what the licensor really had in his mind. As long as we do not have direct insights into the brain of our fellow human beings, this can again only be done by referring to other orally uttered or written words and texts. Therefore, we indeed believe, that it is firstly important to know what the text itself says and means.

Hence, let us prove our position 'bottom up'. Let us firstly show that it is true for the LGPL-v2 – by explicating the license text lingually, then logically, and finally empirically, before we infer the correct understanding. Then let us show that it is also true for the LGPL-v3. And in the end let us show that it is true for all other licenses[312].

3.4.1 Reverse Engineering in the LGPL-v2

The LGPL-v2.1 contains one sentence which literally refers to the issues of *reverse engineering*:

> *"[...] you may [...] combine or link a 'work that uses the Library' with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and* reverse engineering *for debugging such modifications."*[313]

Hereinafter, we will sometimes denote these lines by the term *LGPL2-RefEng-Sentence*.

311) Eben Moglen, eMail to ftf-legal-bounces@fsfeurope.org, 2015-03-06

312) analysed by the OSLiC: → p. 71.

313) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp., §6.. The first ellipse in this citation – notated by the string '[...]' – refers to the phrase "As an exception to the Sections above,", the second to the phrase "also". These words together want to indicate, that the LGPL offers its §6-way-of-distribution as an exception to the intended default way of distributing such a Library. So, the nature of the extraordinary way itself is not affected by this hint. Thus, we feel free to erase this contextual link.

3.4.1.1 Linguistical Clarification

For fulfilling our rule, to read the text strictly and deduce our interpretations reasonably, let us firstly only highlight the syntactical conjunctions for simplifying the understanding:

> *"[. . .] you may [. . .] combine* ***or*** *link a 'work that uses the Library' with the Library to produce a work containing portions of the Library* ***and*** *distribute that work under terms of your choice,* ***provided that*** *the terms permit modification of the work for the customer's own use* ***and*** reverse engineering *for debugging such modifications."*[314]

It is evident that the conjunction *'provided that'* is splitting the sentence into two parts: you are allowed to do something *provided that* a condition is fulfilled. Additionally, both parts of the sentence – the one before the conjunction *'provided that'* and the part after it – are syntactically condensed embedded phrases which also contain subordinated conjunctions and elliptical constructions[315]. These syntactical interconnections must be disbanded:

Let us firstly **dissolve the syntactical compression before the conjunction *'provided that'***: It is established by using the two other conjunctions *and* and *or* and introduced by the subordinating phrase *you may [. . .]*. Unfortunately, from a formal point of view, one can read the phrase *you may (X or Y and Z)* as two different groupings: either as *you may ((X or Y) and Z)* or as *you may (X or (Y and Z))*.

But, fortunately, we know from the semantic point of view that speaking about *"[. . .] combining* ***or*** *linking [. . . something] to produce a work containing portions of the Library"* denotes two different methods which both can *join* the components *"[. . .] to produce a work containing portions of the Library"*. So, let us – only for a moment[316] – simply replace the string *"combine or link"* by the string *"*join"*[317]. This reduces the syntactical structure of the sentence back to the simple phrase *you may (W and Z)* in which *W* stands for *(X or Y)*.

Now, we can directly state that the phrase *you may (W and Z)* itself is a condensed version of the explicit phrase *(you may W) and (you may Z)*.

Finally we have to note, that the phrase before the conjunction *'provided that'* contains also a linguistic ellipsis[318]: It says that you may *join the components

[314] cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp., §6, emphasis KR..
[315] cf. http://en.wikipedia.org/wiki/Ellipsis_%28linguistics%29, wp.
[316] Later on we will re-insert th original phrase!
[317] When the LGPL and the GPL were initially defined, the C programming language was the predominant model of software development. Knowing this method eases the understanding of these licenses. Thus, it is not totally wrong to take this token *join also as a curtsey to the C programming language.
[318] cf. http://en.wikipedia.org/wiki/Ellipsis_%28linguistics%29, wp.

"to produce **a work containing portions of the Library and** distribute **that work** under terms of your choice". With respect to the English grammar, we may conclude that the second term *that work* refers back to the previously introduced specification of *a work containing portions of the Library*: if a complete phrase has just been introduced explicitly, then the English language allows to reduce its next occurence syntactically while its complete meaning is retained. Hence, conversely, we are allowed to unfold the reduced form to restore the complete phrase.

So – overall – we may understand the phrase before the conjunction *'provided that'* as a phrase with the structure *(you may W) and (you may Z')*:

> ***((**you may [...] *)* *join *a "work that uses the Library" with the Library to produce a work containing portions of the Library**) and (**you may [...] distribute that work containing portions of the Library under terms of your choice**))*** **provided that** [...]

Theoretically, a reader could reject our first dissolution of the LGPL2-RefEng-Sentence. But for reasonably denying our interpretation he has to deliver other resolutions of the lingustic elliptical subphrases or other dissolvations of the conjunctions. Fortunately, it seems to be evident that such attempts must violate the English grammar.

Let us secondly **dissolve the part after the conjunction *'provided that'***: With respect to the subordinated conjunction *'and'*, the subphrase *the terms permit* syntactically refers to both, the *modification* and the *reverse engineering*: An embedded conjunction *'and'* allows to use a more stylish grammatical compaction. So, it should be clear, that saying

> **provided that** *the terms permit modification of the work for the customer's own use* **and** *reverse engineering for debugging such modifications*

means

> **provided that** *the terms permit **(** modification of the work for the customer's own use* **and** *reverse engineering for debugging such modifications**)***

and is totally equivalent to the sentence

> [...] **provided that** ***((**the terms permit modification of the work for the customer's own use**)* **and** ***(**the terms permit reverse engineering for debugging such modifications**))***.

We believe that there is no other possibility to understand this part of the LGPL2-RefEng-Sentence with respect to the rules of the English language. Nevertheless,

this is a next point where our reader may formally disagree with us. If he really wants to object our dissolution, he must deliver another valid interpreation of the scope of the conjunction *and* or he must deliver another resolutions of the linguistic ellipsis. But we reckon, that one can not reasonably argue for such alternatives.

Finally, there are other deeply embedded ellipses, which need to be resolved as well:

1. In the part before the splitting conjunction *'provided that'* we already had to expand the abridging *'that work'* by its intended explicated version *'that work containing portions of the Library'*. In the part after the splitting conjunction the first subphrase also contains the term *'the work'*. Formally, this term can either refer to *'the work that uses the library'* as one of the components which are joined, or it can refer to *'the work containing portions of the Library'* as the result of joining the components. We decide to constantly dissolve the elliptic abridgement by the phrase *'the work containing portions of the Library'*.
2. The first clause of the part after the splitting conjunction *'provided that'* talks about the purpose of "permitting modification of the work" which we just had to unfold to the phrase *'permitting modification of the work containing portions of the Library'*. The second clause talks about the purpose of "permitting reverse engineering": it shall support the "debugging [of] such modifications". The pronoun *'such'* indicates that the word *'modifications'* refers back to the just unfolded phrase *modification of the work containing portions of the Library*. So, even the second sentence has to be expanded to that explicit phrase.
3. Finally and only for being complete, we also have to unfold the clause "the terms" to the form which is predetermined by the first referred instance "the terms of your choice"

So – overall – we are allowed to rewrite the LGPL2-RevEng-Sentence in the following form, namely without having changed its meaning[319]:

```
( ( you may
       *join a work that uses the Library with the Library
        to produce a work containing portions of the Library )
  AND
  ( you may
        distribute that work containing portions of the Library
        under terms of your choice
) )
```

[319] Recollect that '*join' still stands for 'combine or link'.

```
PROVIDED THAT
( ( the terms of your choice permit
        modification of the work containing portions of
        the Library for the customer's own use )
  AND
  ( the terms of your choice permit
        reverse engineering for debugging modifications
        of the work containing portions of the Library
) )
```

At this point we must recommend all our readers to verify that this 'structurally explicated presentation' does exactly mean the same as the initially quoted LGPL2-RefEng-Sentence. We are now going to discuss some of its logical aspects by some formal transformations. For accepting these operations and linking the results back to the original LGPL2-RefEng-Sentence, it is very helpful to know that one already has accepted the equivalence of this explicated form and the more condensed original version. For reviewing the equivalence the reader could – for example – ask himself which of our rewritings are wrong, why they are wrong and which alternatives can reasonably be offered for solving the syntactical issues which disposed us to chose our solutions. Again, we ourselves – of course – are profoundly convinced that both versions are completely equivalent.

3.4.1.2 Logical Clarification

For simplifying our discussion let us now replace the meaningful terminal phrases of our form by some logical variables:

Γ :- (you may *join a work that uses the Library with the Library to produce a work containing portions of the Library)

Δ :- (you may distribute that work containing portions of the Library under terms of your choice)

Φ :- (the terms of your choice permit modification of the work containing portions of the Library for the customer's own use)

Σ :- (the terms of your choice permit reverse engineering for debugging modifications of the work containing portions of the Library)

Θ :- Γ *and* Δ

Ω :- Φ *and* Σ

Based on these definitions, we can syntactically reduce the LGPL2-RefEng-Sentence to the formula (Γ *and* Δ) *provided that* (Φ *and* Σ) or – even shorter – to (Θ *provided that* Ω).

Now, we have to clarify the meaning of the conjunction *'provided that'*:

Obviously, *provided that* means something like *under the condition that*. So, one might try to take this conjunction as another more stylish version of the common *if(...)then(...)*-formula, sometimes also identified as a (logical) implication[320]. Thus, we have to consider the process of sequencing the linguistic form into a logical formula: if we indeed take the conjunction *provided that* as another form of the logical implication, it is not evident, which part of the linguistic sentence must become the premise, and which the conclusion: Does *(Θ provided that Ω)* mean *(if Θ then Ω)* or *(if Ω then Θ)*?

Apparently, *provided that* wants to establish something like a precondition. So, one might conclude that (Θ *provided that* Ω) means *(if Ω then Θ)* or – more logically notated – $((\Phi \wedge \Sigma) \rightarrow (\Gamma \wedge \Delta))$. If this interpretation is adequate, it must of course fulfill the intended purpose of the corresponding LGPL-v2-section, which wants to regulate the distribution of works containing portions of LGPL libraries.

For facilitating the understanding of our argumentation, let us first check whether this logical interpretation of the linguistic conjunction fits the purpose of the LGPL – by unfolding the slightly reduced version $(\Sigma \rightarrow \Delta)$ back to the corresponding verbal form:

> ***if*** *(* *[...] the terms permit reverse engineering for debugging modifications of the work containing portions of the Library,* *)* ***then*** *(* *[...] you may distribute that work containing portions of the Library under the terms of your choice.)*

Now we can better see the problem: An implication as a whole is false only if the premise is true and the conclusion is false. In all other cases it is true. Especially, it is true, if the premise is false: If the premise is false, then the truth value of the conclusion does not matter in any sense. Thus, if we take this implication as a rule, which shall determine our behaviour, then this implication only supports us, if we already have decided to permit reverse engineering. In this case the rule successfully tells us that we are allowed to distribute the work containing portions of the Library. But from the converse decision that we will not permit reverse engineering, follows nothing - because a false premise does not influence the truth value of the conclusion. Especially, the rule does not tell us that we may not distribute the work containing portions of the Library. So – from the viewpoint of the formal logic – this translation of the original conjunction *'provided that'* says, that if the terms of your own license do not permit reverse engineering for debugging modifications of the work containing portions of the Library[321], then

[320] Actually the logical implication and the computational if-then-construct are not equivalent. Fortunately, we later on can show, that in the context of this discussion the difference can be ignored.

[321] The premise is false.

you may or may not distribute that work containing portions of the Library under the terms of your choice[322]. Hence, we must state that this interpretation does not fulfill the purpose of the LGPL-V2: if reverse engineering is not allowed, the distribution of the work containing portions of the Library is not regulated. We have to conclude, that this sequencing the LGPL2-RefEng-Sentence as a logical implication is wrong.

But we deduced this consequence from a slighty reduced form of the LGPL2-RefEng-Sentence. Thus, we still have to ask, whether we have to derive this conclusion also on the base of the completely unfolded formula $((\Phi \wedge \Sigma) \rightarrow (\Gamma \wedge \Delta))$? The answer is yes: the premise $((\Phi \wedge \Sigma)$ contains a logical conjunction. So the truth value of the whole premise depends on the truth value of each of its terminal statements, particularly on that of the statement Σ: If we decide not to permit reverse engineering, then the premise as whole is false, regardless we forbid or allow modifications. Consequently, the premise does not influence the truth value of the conclusion. So, there is no way, to conclude that we have to allow or that we do not have to allow reverse engineering. Hence we can transfer our result, deduced for the slightly reduced formula to the unfolded complete formula: assuming that (Θ *provided that* Ω) means *(if* Ω *then* Θ) is wrong.

So, let us test the other combination. Let us ask, whether *(*Θ *provided that* Ω*)* means *(if* Θ *then* Ω*)* or – more logically notated – $((\Gamma \wedge \Delta) \rightarrow (\Phi \wedge \Sigma))$. If we again for a moment focus on the reduced version $(\Delta \rightarrow \Sigma)$ and dissolve our replacements, then we get back the rule:

> ***if*** *(* *[...]* *you may distribute that work containing portions of the Library under the terms of your choice,* *)* ***then*** *(* *[...]* *the terms permit reverse engineering for debugging modifications of the work containing portions of the Library.* *)*

Now we can see, that this version perfectly regulates the distribution of works containing portions of LGPL libraries: If we are allowed to do so or – in other words: if we are compliantly distributing works containing portions of LGPL libraries[323], then we have to permit reverse engineering[324]. This follows from applying *Modus Ponens* to the implication[325]. And if we do not permit reverse engineering[326], then we are not allowed to distribute works containing portions of LGPL libraries[327]. This follows from applying *Modus Tollens* to the implication[328]

[322] The truth value of the conlusion is undetermined by the rule.
[323] The premise is true.
[324] The conclusion must be true, too!
[325] A true premise evokes a true conclusion based on the given truth of the implication / rule itself.
[326] The conclusion is false.
[327] The premise must be false, too!
[328] A false conclusion evokes a false premise based on the given truth of the implication / rule

But – again – we have to consider that we have deduced this consequence from a slighty reduced version of our LGPL2-RefEng-Sentence. Thus, we still have to show that our result also holds for the completely unfolded formula $((\Gamma \wedge \Delta) \rightarrow (\Phi \wedge \Sigma))$: If we want to distribute works containing portions of the Library which have been produced by joining the Library and the work using the Library[329], then our terms must permit the modification *and* reverse engineering of the distributed product[330]. And if we do not allow its modification *or* reverse engineering[331], then we do not compliantly distribute works containing portions of the Library which have been produced by joining the Library and the work using the Library[332] Thus, we may generally state, that the logical explication $((\Gamma \wedge \Delta) \rightarrow (\Phi \wedge \Sigma))$ perfectly regulates the distribution of works containing portions of LGPL libraries.

Based on this clarification, we can reasonably replace the more stylish conjunction *'provided that'* by its more known equivalent *'implication'*[333], which we indicate by the commonly used character for a logical implication, the sign '$\rightarrow$':

\# Θ provided that Ω

$\equiv \Theta \rightarrow \Omega$

$\equiv (\Phi \wedge \Sigma) \rightarrow (\Gamma \wedge \Delta)$

$\equiv$ `( ( [Φ] you may`
`*join a work that uses the Library with the Library`
`to produce a work containing portions of the Library )`
$\wedge$
`( [Σ] you may`
`distribute that work containing portions of the`
`Library under terms of your choice`
`) )`
$\rightarrow$
`( ( [Γ] the terms of your choice permit`
`modification of the work containing portions of`
`the Library for the customer's own use )`

itself.

329) Premise is true.

330) Conclusion must become true by Modus Ponens.

331) Conclusion is false.

332) Premise must become false by Modus Tollens.

333) Here we can also see, that the difference between the if-then-command as part of a procedural computer language and the logical implication does not influence our results: In the context of a procedural if-then-command the truth of the premise triggers the execution of the conclusion. In our discussion, this aspect is totally covered by the Modus Ponens derivation of the logical interpretation. And the Modus Tollens derivation of the logical interpretation on the other side does not play any role in a procedural if-then-command. So, it was the right decision to understand the LGPL2-RefEng-Sentence logically and not as procedual command.

```
    ∧
    ( [Δ] the terms of your choice permit
          reverse engineering for debugging modifications
          of the work containing portions of the Library
) )
```

3.4.1.3 Empirical Clarification

We can now simplify this formula once more by considering some empirical facts and explicating some underlying understandings:

The first sentence Φ explains that the *work that uses the Library* and the used *Library* itself together are joined and thereby transformed into a *work containing portions of the Library*. So, formally, one might ask, whether this newly generated *work containing portions of the Library* also still *uses the Library*?

Unfortunately, it is empirically possible, that such a process for combining the two components could (a) copy all original portions of the library into a something like a 'dead end section' of the program where they are never excuted, and could (b) replace all original portions of the library by functionally equivalent portions of any other library. Thus, the resulting *work containing portions of the Library* would indeed still contain portions of the Library, although it would not use it any longer. And because of this possibility, we are not allowed to say, that every work containing portions of a library also uses the library[334].

But, fortunately, the normal computational process of *combining and linking a work that uses the Library with the Library to produce a work containing portions of the Library* inherently preserves the utilization of the joined library: It is the general purpose of a software library to offer functions and/or data (structures) for really being used by applications. And vice versa, software developers refer to a specific library because they prefer its service: They use readily prepared libraries (or classes or anything else) because they want to simplify their own work while they conserve the quality level of their work. Thus, they chose a library based on the assertion, that the standard compiling and linking process guarantees, that indeed the chosen library is used (and not secretly substituted by a mysterious 'equivalent'). With respect to this praxis of programming we are allowed to say that a *work containing portions of the Library* which has been **built by the normal development processes** of combining, compiling, and linking source and object files, indeed also uses the intended library.

Now, we are able to consider an empirical correlation between the first sentence Φ and the second sentence Σ:

334) ... even if we think that this is a really silly way to organize the joining process!

It seems to be evident, that we must already have done Φ, in other words: that we must already have **joined – respectively: combined or linked – a work that uses the Library with the Library to produce a work containing portions of the Library*, if we are going to compliantly *distribute that work containing portions of the Library under terms of your choice*. Or briefly spoken: It seems to be conclusive that Σ ***empirically*** *implies* Φ[335].

But is this conclusion correct? Let us check this statement by assuming the opposite: If the contrary was true, there had to exist a *work containing portions of the Library* which had been gained without having linked or combined the work and the Library in any sense. But from the inference above we already know that *works containing portions of the Library*, which have been produced by the standard computational processes of *combining and linking a work that uses the Library with the Library*, indeed also *use the Library*. Thus, it would be self-contradictory to talk about a *work containing portions of the Library*, which was produced by the standard combining and linking processes, and similarily to state, that exactly this work is not combined with the library in any sense. And from a proof by contradiction we may infer the truth of the logical opposite:

So, with respect to the meaning of *being standardly combined or linked with*, we may now say, that

- it is necessarily true that a computional work, which is standardly produced on the base of *a work that uses the Library* and *the Library* and which therefore literally contains more or less *portions of a library*, indeed uses the *the Library* and *is* therefore *combined with the library*.
- Σ[336] empirically implies Φ[337] (in the standardized world of software development), because Φ must ever have been executed when Σ is going to be realized.

Thus, we can now reduce the LGPL2-RefEng-Sentence to its real core, the LGPL2-RefEng-Rule:

```
(   [Σ] you may
        distribute (a) work containing portions of the
        Library[338] under terms of your choice )
→
( ( [Γ] the terms of your choice permit
```

[335] but not vice versa.

[336] distributing *a work that uses the Library and contains portions of a library*

[337] A work that uses the Library has been *joined with the Library to produce a work containing portions of the Library

[338] which previously has been prepared for being distributed by standardly combining and linking the work that uses the Library with the Library in a way that this prepared work indeed also uses the Library

```
        modification of the work containing portions of
        the Library for the customer's own use )
  ∧
  ( [Δ] the terms of your choice permit
        reverse engineering for debugging modifications
        of the work containing portions of the Library
) )
```

This is indeed the essence of the LGPL2-RefEng-Sentence. It logically explains us that we have to *allow reverse engineering* and modification of a *work containing portions of the Library* if we distribute it (Modus Ponens) and that we are *not allowed to distribute a work containing portions of the Library*, if we do *not allow* its modification or *reverse engineering* (Modus Tollens).

Thus, for applying this rule correctly, we now only must know whether a work indeed contains portions of the Library or not.

3.4.1.4 Final Conclusion

Unfortunately, there are more than one software developing scenarios, which must be considered for answering this question in detail. We see three general types of developing computer software:

1. You can produce software by using script languages. Source files which contain script language commands are distributed and executed by an interpreter without priorly being transformed into another 'more' machine specific language.
2. You can develop software by using languages which are designed for being compiled into a machine independent bytecode. Later on, this independent bytecode is executed by a machine specific virtual machine.
3. You can write traditional software files. Sometimes, these files are remastered by a preprocessor before the real process starts. The traditional sources themselves or the output files of the preprocessor are then compiled and linked as machine specific binary file(s).

You may take 'php' is an example for the first environment, 'Java' an example for the second, and 'C/C++' an example for the third.

Fortunately, the nature of these environments simplifies the answer to the question under which conditions the work using the Library contains portions of the Library:

3.4.1.4.1 Distributing works with manually copied portions of the Library evokes the copyleft effect: Manually copying code from the sources of the

Library into the overarching work that uses the Library, is not the standard way of combining both components, neither in the world of script programming, nor in the world of bytecode programming, nor in the world of programming machine specific code:

Normally, the work which uses the Library is joined to the intended Library by an include statement, an input statement, an import statement, a package statement, or anything else. These *join-statements are inserted into the code of the work. They denote the file(s) which deliver(s) the used functions, methods, classes, or data. It is an integrated feature of the normal development tools that inserting such *join-statements does not directly augment the work using the library by some code of the Library: The development processes are designed to offer an automatic augmention as part of the standard compilation which is started after the actual development loop has been terminated.

Nevertheless, developers can circumvent these standard methods for using a Library. Technically, they can directly copy code from the Library into their own work. Consequently, these manually copied extensions of the code will be compiled and/or executed together with the 'own' code of the work. Thus, it is clear that in this case the work that uses the Library already contains portions of the Library, particularly before the normal *join-processes of the environment are executed.

Hence, if you are going to distribute works that contain literal copies of the Library source code, then you have to allow reverse engineering, even if they have already been compiled (but still not linked) on the base of such augmented files[339].

[339] This directly follows from the LGPL2-RefEng-Rule by Modus Ponens. But nevertheless, one might reply here, that even the result of manually copying code from the Library to the work using the Library is covered by the limits of tolerance, introduced by the LGPL-v2-§5. Formally, this argument seems to be appropriate. And indeed, also we have to consider these limits of tolerance later on. But in the context of copying code from the Library into the work manually, a closer look reduces its impact. You have to discriminate three cases:

1. Developers can **manually copy / transfer some or at most all elements of the Library header files into the code of the work** which the preprocessor itself would copy / transfer into that code automatically. But developers will not do that. Some simple include commands would cause the same effect. And developers want to save resources, especially their own working time. So, why should they manually do what they can delegate to the standard development process. Thus, it is reasonable to assume that developers, who nevertheless copy portions from Library into their work, do not want to repeat the service of the preprocessor manually, but to transfer more than only these elements. Hence, it is reasonable to assume, that their work is covered by the LGPL2-RefEng-Rule.

2. Developers can **manually copy / transfer more than only the elements of the Library header files from the Library sources into the code of the work using the Library and they can nevertheless let the work being linked to the Library**. But again developers will not do that, because – again – some simple include and linking commands would cause the same effect. So it is reasonable to

But, if we manually copy code from the Library to our work using the Library, we also have to consider that the LGPL-v2 directly regulates this kind of using the Library: It says, that "you may modify your copy or copies of the Library or any portion of it [...] provided that you also [...] cause the whole of the work to be licensed [...] under the terms of this License"[340]. Thus, there are strong arguments for the proposition, that the LGPL causes the copyleft effect in case of literally copying code from the Library into the work using the Library: The code of the work using the library has to be made accessible, as well.

So, overall, we might say, that 'manually' copying code from an LGPL-v2 Library into a work using that Library as a bypass of the standard software combining processes and distributing the result indeed requires to additionally permit its reverse engineering – even if this permission is probably not very important for the recipient, because he probably must have a direct access to the code.

3.4.1.4.2 Distributing scripts does not need reverse engineering: Computer programs written in a script language are distributed as they have been developed. They are not transformed into another kind of code[341]. The interpreter takes the script file as it is and directly executes it. Thus, there is no special technique of reverse engineering for understanding these kind of software: you can directly read it if you know the script language.

So, again, we might conclude, that a script using a script Library perhaps requires to permit its reverse engineering – but probably this permission is not very important for the recipient, because he can directly read the code.

3.4.1.4.3 Distributing statically combined bytecode requires the permission of reverse engineering: In Java – the prototype for languages which are compiled

start from the premise that copying developers in fact do more than this. Thus, it is reasonable to assume that their work is covered by the LGPL2-RefEng-Rule.

3. Developers can **manually copy / transfer more than only the elements of the Library header files from the Library sources into the code of the work using the Library without linking it to the Library**. This is a reasonable step of work, because it spares the developers to link their work to the Library. But – by definition – such an augmented work contains more elements of the Library than LGPL-v2-§5 tolerates. Thus it is – again – reasonable to assume, that such a work is covered by the LGPL2-RefEng-Rule.

Hence – overall and from a practical point of view – we can indeed say that manually copying code from the Library into the work using the Library requires to allow reverse engineering.

340) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp., §2, escpcially §2c.

341) Java script is often offered as compressed code. Roughly spoken, this means that at first all white space signs have been replaced by blanks and then all rows of blanks have been reduced to at most one single blank. So, even then, the code itself is directly readable and comprehensible – even if only for very sophisticated experts.

to machine independent portable bytecode – each class is compiled as a separate class file. These class files have to be stored somewhere in the classpath. Aside from that, classes can also be collected and distributed in form of packages which then can be used like 'traditional' Libraries. These packages must also be stored somewhere in the classpath. A single class is made known to the work that wants to use it by an import statement which contains the class name; a whole Java library is made usable by integrating a package statement into the code.

The code which follows such import- or package statements, can then use the definitions offered by the classes. It denotes the elements of the classes by the (qualified) names of its public or protected member variables or methods. Thus, – from a strict viewpoint – the code of such a Java work using a Library indeed contains portions of that library, even if these portions are only identifying names or data structures containing identifying names. The Java compilation process which generates the bytecode, preserves these denoting names. It does not replace the referring names by the referred code of the methods and so on. Only just at the end, when the java virtual machine itself tries to execute the work using the Library, it collects all necessary commands of all 'joined' classes.

So, one might tend to argue that answering the question whether a distributed java bytecode already contains portions of the used Library depends on the interpretation whether a denoting identifier of a Library indeed is a portion of the Library. We will discuss this case together with the corresponding C/C++-Case.

But there is another Java specific aspect, which has to be considered as well. As already mentioned, in Java you can also join your work containing the denoting identifiers and the denoted Library by building a new package, which then contains both, the work using the Library and the used Library. Hence, one can say, that this package is quasi statically linked: if you distribute such an integrated package, then you are distributing both components together. Thus, if you distribute a complete package, in other words: a quasi statically linked work containing the work using the Library and (all portions of) the Library, then you have to permit reverse engineering[342].

So, preliminarily we conclude that, with respect to Java programming you (a) have to permit reverse engineering, if you distribute your work using the Library and the Library itself as a (statically linked) integrated package[343] and that (b) in all other cases your obligation to permit reverse engineering depends on the interpretation whether the identifiers declared by a Library are indeed portions of the Library.

Fortunately, we can reasonably decide the issue of case (b) soon.

[342] This directly follows from the LGPL2-RefEng-Rule by Modus Ponens
[343] This follows from the LGPL2-RevEng-Rule by Modus Ponens.

3.4.1.4.4 Distributing statically combined binaries require the permission of reverse engineering: Similar to Java, in C/C++ – the prototype of those languages, which are compiled as machine specific code – a C/C++ Library is also explicitly made known to the work that wants to use it, namely by some include statements. These include statements denote the header files offered by and distributed with the Library. They contain the declarations of those elements which the Library wants to publish. Or briefly worded: the Library contains the definitions in form of code, the header files the corresponding declarations.

The C/C++ code following such include statements can refer to the definitions offered by the Library by using the declarations anounced by the header files. So, again, – from a strict viewpoint – the code of such a C/C++ work using the Library indeed contains portions of the library, even if these portions are only identifying names or data structures published by the header files.

Beyond that conceptual relation, the C/C++ development process finally compiles the work using the library as an object file containing machine specific code. Just as the Java compilation, this process does not replace the referring names by the referred code of the Library; it still preserves the denoting names. The resulting file, which has been compiled into machine specific code, but still contains the denoting identifiers, is also known as 'object code file'.

The C/C++ compilation process is (mostly) managed by a make file, which is executed by the make command[344]. This development tool calls the compiler for each source file, makes known the directories which contain the compiled target object files, and finally calls the linker. The linker recursively scans the compiled object files and replaces each embedded identifier by a truly executable jump command into that set of Library commands which are denoted by the identifier and which shall be executed as part of the work using the Library. So, only at the end, the linker collects all necessary commands of all 'joined' object files and Libraries and produces the really executable work.

But – notwithstanding the above – the linker can either be called as integrated step of the development process itself, or the linker can be called separatedly, especially on another machine. In the first case, the development process generates a *statically linked executable* which already contains all necessary portions of all used Libraries. In the second case, the development process generates a *dynamically linkable program* by collecting the (set of) still unlinked object code file(s) as a distributable package. Thus, if you distribute a statically linked executable, it definitely contains 'portions' of the library; if you distribute a dynamically linkable program you have to decide whether the embedded identifying names of a Library have indeed to be interpreted as portions of the Library.

[344] Sometimes there additionally exists a complete meta environment which generates such make files. The GNU build system for example offers a complex set of configure scripts and make file templates (cf. http://en.wikipedia.org/wiki/GNU_build_system, wp.).

Unfortunately, we still have to consider a little complication, based on the nature of the a C/C++ development process: contrary to the Java development environment, a C/C++ development process inherently uses a preprocessor engine. This engine takes the header files delivered by the Library, verifies the syntactically correct use of the Library and can indeed replace some tokens of the work using the Library by commands and/or lines from the Library. This technique is known as *inline functions* or *macros*. They have been invented for those cases where expanding the stack of commands during the compilation by a real function call is more expensive than writing the embedded commands of the function more than one time into the whole code. Hence, in the C/C++ development process the compiled object files can indeed contain more than only the referring names which denote portions of the Library: beside the denoting identifiers, they can also already contain real, functionally relevant portions of the Library.

Thus, – again and similar to Java compilation – we may conclude, that with respect to C/C++ programming you (a) have to permit reverse engineering, if you distribute your work using the Library together with the Library as a statically linked program[345] and that (b) in all other cases your obligation to permit reverse engineering depends on the interpretation whether the used identifiers or dissolved inline functions and macros, which have been declared by the Library and which therefore have automatically and standard conformably been embedded into an object file, are indeed portions of the Library.

Obviously, it is time to answer this crucial question:

3.4.1.4.5 Distributing dynamically combinable bytecode and linkable object code does not require the permission of reverse engineering: Of course, there is only one instance that can answer the question whether identifiers and dissolved inline-functions or macros, which are – according to the development standard – embedded into a work using the Library, indeed are portions of the Library. This instance is the LGPL-v2 itself. And – fortunately – this license supports us in a very clear way to answer this question, even if not by its §6 which deals with the reverse engineering, but by its §5:

The LGPL simply specifies that "linking a 'work that uses the Library' with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library) [...]" and that "the executable is therefore covered by this License"[346]. Additionally, it talks about compiled, but still unlinked "object files", which therefore are not executables. Such an unexecutable "object file" – for example that of the "work using the Library" –, which "[...] uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length)" shall practically not be

[345] This follows from the LGPL2-RevEng-Rule by Modus Ponens.

[346] cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp. §5.

covered by the license of the Library, because "[...] the use of the object file is unrestricted regardless of whether it is legally a derivative work"[347] - as long as it does not exceed the given limits.

Obviously, the answer of the LGPL to our question is this: (a) yes, such object files containing names and snippets offered by the used Library, could contain portions of the Library. But it is not necessary to clarify the details, because (b) – up to a specific limit of sizes – these kind of 'little' portions being embedded into the object file by the standard compilation processes do not evoke any requirements: they especially do not evoke the obligation to allow reverse engineering. In other words: these little portions of a Library which are embedded by the standard development process and which do not contain more than the specified size of code may be regarded as another type of portions compared to the normal, real portions which indeed evoke the obligation to allow reverse engineering. From the viewpoint of the LGPL, they are *pseudo portions* of the Library, because they do not restrict the containg object file in any respect.

So, from the LGPL-RevEng-Rule we can now indirectly conclude, that distributing dynamically linkable or combinable bytecode or object code files which contain "only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length)" being delivered by a Library does not require to allow reverse engineering[348].

Unfortunately, there might be a practical objection which seems to disturb our simple result: For applying this rule correctly, we apparently have to assure that a compiled work that uses the Library but is still not *joined to it, indeed has only been expanded by "small macros or small inline functions (ten lines or less in length)". Thus, seemingly, we have to study all header files of all used Libraries in detail, if we want to compliantly distribute a work using a Library without permitting reverse engineering. This could be a lot of work – up to a bulk which practically can not be managed.

Fortunately, there is a simple solution for this challenge, a rule of thumb, based on the principle "trust the upstream"[349]:

The Library developers of course publish the header files or the public members and functions of the classes in exactly that form they want these elements to be

[347] cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp. §5.

[348] From the decision not to allow reverse engineering follows by Modus Tollens applied to the LGPL2-RevEng-Rule, that the distribution of the work using the Library must not contain real portions of the Library. From LGPL-v2-§5 and the limit of the standard proccesses follows that here the work using the Library does not contain normal, real portions. So, we know, that this case is not covered by the LGPL2-RevEng-Rule and thus we are allowed to distribute a work using the Library without allowing its reverse engineering.

[349] On the ELLW 2013, we were told about this principle for the first time. We do not know, whether Armijn Hemel invented it. But we can respectfully affirm that he has persuasively explained the spirit and purpose of the principle "trust the upstream".

used. And they want their Library to be used as an LGPL library, otherwise they would have chosen another License. So, they wish that improvements of the Libraries shall be made accessible as well, but that the works using the Library shall not necessarily be published in form of source code[350]. Thus, as long as we use a Library exactly in that form, the original authors have published, as long as we download the Library from the official repository, and as long as we do not modify the intended interfaces defined and published by the original header and class files, we may justifiably assume that we are using the Libraries just as their copyright owners want them to be used. And thus, – in other words: as long as we trust the upstream – we might assume that the header and class files of our Libraries fit the restrictions of the LGPL-v2.

3.4.1.4.6 LGPL-v2 compliance with or without permitting reverse engineering:

Now, we have reached our target. Our last clarification can directly be applied to the both open cases: to the case of distributing Java bytecode as well, as to the case of distribution C/C++ object code. We now know, that the LGPL-v2 wishes, that not all portions of a Library covered by a work using the Library, trigger the permission of reverse engineering. And we now know that the limits – given by the LGPL-v2-§5 – up to which such pseudo portions indeed do not trigger the obligation to permit reverse engineering, are respected, if we use *'upstream approved'* C/C++ and Java libraries in standard development environments. Thus, we indeed finally may conclude, that the LGPL-RevEng-Sentence

> *"[. . .] you may [. . .] combine* ***or*** *link a 'work that uses the Library' with the Library to produce a work containing portions of the Library* ***and*** *distribute that work under terms of your choice,* ***provided that*** *the terms permit modification of the work for the customer's own use* ***and*** reverse engineering *for debugging such modifications."*[351]

means 'nothing else' than

- *With respect to a LGPL-v2 licensed Library, you are not required to allow reverse engineering, if you [A] develop your work using the Library, on the base of a standard version of the Library containing the interfaces as the original developers have designed it, if you [B] compile your work using this Library, as a discrete (set of) dynamically linkable or combinable file(s), if you [C] use only the standard compilation methods which preserve the upstream approved interfaces*[352]*, and if you [D] distribute the produced unlinked object code or bytecode files before they are linked as an executable.*

[350] The meaning of the weak copyleft.
[351] cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp., §6, emphasis KR..
[352] and which therefore do not to exceed the LGPL-v2 limits!

- *In all other cases of distributing a work using such a Library, you are required to allow reverse engineering of the work using this Library – especially, ...*
  - *if you distribute the work using the Library and the Library together as a statically linked program or as an integrated package containing both parts, the work using the library and the Library itself*[353].
  - *if you distribute a work containing manually copied portions of the Library.*

3.4.1.5 Final Securing

So far, we have done a lot of work: At first, we unfolded and dissolved some stylisch condensed formulations of the original LGPL2-RevEng-Sentence by their linguistically explicit version. At second, we explicated the logical structure of the sentence. At third, we empirically carved out the real meaning of the sentence. And finally we mapped the triggering part of that rule to some verifiable facts. Indeed, a lot of work for understanding only one sentence correctly[354]. So, it is a good securing to verify that the derived result fits the spirit and the goals of the LGPL-v2 perfectly.

For that purpose, let us fist discuss a little (semi-) official article – written by David Turner and published by the FSF[355] – which deals with the compliant use of LGPL licensed Java libraries. Turner refers to the "FSF's position" which - as he says - "[...] has remained constant throughout":

> *"[...] the LGPL works as intended with all known programming languages, including Java. Applications which link to LGPL libraries need not be released under the LGPL. Applications need only follow the requirements in section 6 of the LGPL: allow new versions of the*

[353] This holds also if you distribute a script language based program or package, notwithstanding the fact, that one does not need the permission of reverse engineering to understand script language based applications.

[354] Here, some readers might ask why the original authors have encapsulated their clear ideas in such a sophisticate sentence. Here are two answers: First, this question is practically irrelevant: The authors of the LGPL-v2 did, what they have done. And many developers have already licensed their works under the terms of the LGPL-v2. Thus, we simply have to live with the results – just until the last software being published under the terms of the LGPL-v2 is relicensed by a better version. Probably this won't happen during our life time. Secondly, we appreciate the foresight of the LGPL-v2 authors. They wrote a license which have successfully worked for more than twenty years. They chosed a formulation which had also to cover 'uninvented' techniques. So, it is not so surprizing, that we – today – have to do a lot of work to understand all details the original authors want to be understood.

[355] cf. *Turner, David*: The LGPL and Java; 2004 ⟨URL: http://www.gnu.org/licenses/lgpl-java.en.html⟩ – reference download: 2015-02-09, wp..

> *library to be linked with the application; and allow reverse engineering to debug this."*[356]

Then he describes, that Java libraries are "[...] distributed as a separate JAR (Java Archive) file" and that applications "[...] use Java's 'import' functionality to access classes from these libraries". Moreover, he also explains, that the process of compilation "creates" and integrates "links" into the compiled application which let become the application a "derivative work" of the library. Finally he states, that not only the LGPL permits to distribute such derivative works, but that "[...] it is easy to comply with the LGPL" if one indeed wants to "[...] distribute a Java application that imports LGPL libraries": "Your application's license needs to allow users to modify the library, and reverse engineer your code to debug these modifications."[357]

So, we might state, that even this semi-official article argues very similarly to us. There is only one little phrase in this text which differs a little: Summarizing the "section 6 of the LGPL" by the statement "*[...] allow new versions of the library to be linked with the application; and allow reverse engineering to debug this*" does not consider that the first sentence of the section 6 of the LGPL contains a complex condition. The *LGPL2-RefEng-Sentence* means – as we could prove – that *one may distribute (a)* ***work containing portions of the Library*** *only if one's license permit reverse engineering for debugging modifications*[358]. But – as we could also show – for determining wether an application really contains portions of the Library, one has additionally to consider the limits defined by section 5 of the LGPL[359]: the application's license needs to allow to reverse engineer the application only if it contains more elements of the Library than §5 of the LGPL-v2 has specified as limit.

That our analysis fits the spirit of the LGPL, can also be shown by considering the LGPL directly:

The LGPL-v2 clearly describes its goals. It wants to enable the community to let an LGPL Library "[...] become a de-facto standard". And the LGPL knows, that "to achieve this [goal], non-free programs must be allowed to use the library", because the "[...] permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software". But the LGPL also asserts in this context, that "although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library"[360].

---

356) cf. *Turner*: The LGPL and Java, 2004, wp.
357) cf. id., l.c., wp..
358) → p. 82
359) → p. 88
360) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp., preamble, emphasis KR.

So – as a last check of our derivation – we can analyze, whether our derived result violates this goal. If it does, then we probably made a tremendous fault; if not, then we are allowed to trust in the consistence our analysis:

If you receive a work using the Library in form of a discrete (set of) dynamically linkable or combinable file(s) and if – hence – your provider assumed that the files he delivers will be linked on your target machine which – therefore – has to provide a linker and the the necessary dynamically linkable Libraries, than you systematically have the freedom to replace the dynamically linked Libraries by their updated versions[361]. And as long as the newer versions of the Libraries preserve the defined and declared interfaces, you can do that successfully. That's, what the LGPL-v2 wants to ensure.

In all other cases, you must have the permission of reverse engineering or you have a direct access to the source code. So, you can use the corresponding tools and techniques to replace the embedded version of the Library by a newer version; especially if you have received a statically linked package. Hence, also the second part of our interpretation respects the spirit of the LGPL-v2.

So, finally we can say, everything is fine: The LGPL2-RevEng-Rule – together with the meaning of being a portion of a Library – does not only verifiably exeplicate the meaning of the LGPL2-RevEng-Sentence, but also fits the spirit and the purpose of the LGPL-v2 as it has been announced by its preamble.

3.4.2 Reverse Engineering in the LGPL-v3

Based on our experiences how to successfully carve out the meaning of license text, we can shorten the way to understand the one LGPL3-RevEng-Sentence referring to *reverse engineering*:

> *"You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following [. . .]"*[362]

Reusing our method of disambiguation, we first can exemplify the meaning of the LGPL3-RevEng-Sentence by the following text:

---

[361] In GNU/Linux – for example – you must (only) copy the dynamically linkable new version of the Library into the lib/-directory and replace the existing link by a version pointing to the newer version. Sometimes you should additionally verify the ld.so.conf files and call ldconfig tool.

[362] cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp., §4. The ellipsis at the end of the sentence denotes a set of tasks which we do not listen here for saving resources, but which have to be considered as an integrated part of this sentence.

```
( [Θ :]
  ( You compliantly distribute a Combined Work
    under terms of your choice
    (   (that together effectively, do not restrict modification of
        the portions of the Library contained in the Combined Work)
    AND
        (that together effectively, do not restrict reverse
        engineering for debugging modifications of the portions
        of the Library contained in the Combined Work)
  )  )
  IF
  [Ω :]
  ( you also do each of the following [...])
)
```

But now, a simply executed logical serialization let us running into a problem:

If we serialized *(*Θ *IF* Ω*)* as *(*Ω → Θ*)*, then from not respecting Θ would follow by Modus Tollens, that we are not allowed to realize Ω – in other words: that we may not do even one of the single tasks covered by the ellipsis – which is a silly result.

If we serialized *(*Θ *IF* Ω*)* as *(*Θ → Ω*)* then from doing Θ would successfully follow by Modus Ponens that we also have to do Ω. And from not respecting Ω would successfully follow by Modus Tollens, that we must not do Θ. But unfortunately, we can respect this second interdiction also *by distributing a Combined Work under terms* that restrict modifications and/or reverse engineering (instead of not restricting these techniques) – which, again, is a silly result.

Obviously, a simple serialization based on a intutively unclear reading fails. In fact, the LGPL3-RevEng-Sentence must have a more sophisticated underlying structure. It must be logically serialized in a form, that integrates the requirements, not to restrict modifications and reverse enigneering, as really triggable conditions. Thus, the meaning of the sentence can logically be explicated as the *LGPL3-RevEng-Rule*:

```
( [Σ :]
  ( You compliantly distribute a Combined Work
    under terms of your choice
  )
  →
  (    [Γ :]
     ( the terms of your choice together effectively do
       not restrict modification of the portions of the
```

```
      Library contained in the Combined Work)
    ∧ [Δ :]
    ( the terms of your choice together effectively, do
      not restrict reverse engineering for debugging
      modifications of the portions of the Library
      contained in the Combined Work)
    ∧ [Ω :]
    ( you also do each of the following [...])
) )
```

This LGPL3-RevEng-Rule indeed successfully regulates how to compliantly distribute a Combined Work by telling us,

- that we have to respect Γ, Δ **and** all single parts of Ω, if we distribute a Combined Work compliantly[363].
- that we do not distribute a Combined Work compliantly, if we do not respect one of the requirements Γ, Δ or one of the single parts of Ω[364].

Now, we can directly see, that the LGPLv3 does not enforce us, not to obstruct reverse engineering in all respects! The required reverse engineering is limited to the purpose of supporting the debugging of modifications and focused to the Combined Work containing portions of the Library. In other words: our terms may obstruct other purposes of reverse engineering or may restrict reverse engineering of other forms of our work which which can not be specified as Combined Work or do not contain portions of the Library. Thus, the first crucial question is, what the LGPL-v3 means if it talks about a "Combined Work". The second question is, what the LGPL-v3 specifies as a portion of the Library.

Again, fortunately, the LGPL-v3 answers clearly: "A 'Combined Work' is a work produced by combining or linking an Application with the Library"[365]. From our LGPL-v2 analysis we know the ways how works that uses a Library can technically be linked or combined with the Library:

- Copying code from the Library into the work using the Library[366] causes that the application respectively the work using the Library indeed contains portions of the Library[367].

[363] follows by Modus Ponens. Thus, in this case especially our terms "[...] together effectively **[must] not restrict reverse engineering** for debugging modifications of the portions of the Library contained in the Combined Work".

[364] follows by Modus Tollens. Thus, especially we are not distributing a Combined Work compliantly, if our terms "[...] together effectively **do restrict reverse engineering** for debugging modifications of the portions of the Library contained in the Combined Work".

[365] cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp., §0.

[366] The LGPL-v3 designates the work using the Library as "Application" and defines that it "[...] makes use of an interface provided by the Library [...]" (cf. id., ibid.).

[367] → p. 83

- Combining script language based applications and Libraries may evoke that the resulting application contains portions of the Library. But the details can be neglected with respect to the reverse engineering, because script code is distributed as it has been developed and can therefore directly be understood[368].
- Combining java classes and libraries as integrated quasi statically linked packages causes, that the resulting package already contains all functionally necessary code of the Library[369].
- Compiling java classes without combining them with the referred Library classes causes, that the compiled classes at least contain identifiers having been declared by the Library[370].
- Combiling C/C++ files or classes and linking them with the referred Libaries statically causes, that the resulting executable indeed contains all functional relevant code of all used Libraries[371].
- Combiling C/C++ files or classes without linking them to the referred Libaries causes, that the resulting object file can dynamically be linked on another machine and contains identifiers offered by the Library and sometimes some functional code injected by dissolving some inline functions or macros[372].

So – overall – the situation is this: The LGPL3-RevEng-Rule tells us that we have to allow reverse engineering of the portions of the Library contained in the Combined Work. The LGPL3 additionally specifies, that a Combined Work is simply the result of technically combining the work using the Library (the application) and the Library. Finally the praxis tells us, that (a) combining both components statically indeed causes that the resulting Combined Work contains portions of the Library[373], and that (b) we – in case of preparing the both parts as dynamically combinable components – still have to clarify whether the resulting work already contains portions of the Library.

Just as the LGPL-v2, the LGPL-v3 supports us to answer this question by its §3 whose linguistic conjunctions we thoroughly have to consider:

> *The object code form of an Application may incorporate material from a header file that is part of the Library.* ***You may convey*** *such object code under terms of your choice,* provided that, *[* ***if*** *the incorporated material is* ***not*** *limited to numerical parameters, data structure layouts*

368) → p. 85
369) → p. 85
370) → p. 88
371) → p. 87
372) → p. 88
373) So, it is triggering the LGPL3-RevEng-Rule.

> *and accessors, or small macros, inline functions and templates (ten or fewer lines in length)], **you do both** of the following: **a)** Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License. **b)** Accompany the object code with a copy of the GNU GPL and this license document]*[374].

The first sentence of this paragraph tells us that he is dedicated to object files which are compiled and not linked to the used Library, but which nevertheless can contain portions of the Library. The second sentence regulates the distribution of such object files and can be logically serialized:

```
( [Λ :]
  ( You compliantly distribute object code [incorporating
    material from the Library] under terms of your choice )
  →
  [Ξ :]
  ( [ω :]
    ( the incorporated material is not limited to numerical
      parameters, data structure layouts and accessors, or
      small macros, inline functions and templates
      [ten or fewer lines in length] )
    →
    ( [α :] ( you do [a] ...] )
    ∧ [β :] ( you do [b] ...] )
) ) )
```

We see, that this LGPL3-sentence concerning the distribution of object files contains a main rule $(\!(\Lambda \to \Xi)\!)$ and that the conclusion Ξ itself has the form of an embedded sub rule $(\!(\omega \to (\ \alpha \wedge \beta)\!)$.

Firstly, the main rule enforces us to respect the sub rule if we want to distribute the object code compliantly[375]. Secondly, the main rule tells us that we do not distribute the object code compliantly if we do not respect the sub rule [376].

We have two ways to respect the sub rule, and one way not to respect it:

- If the object code contains more and/or larger elements of the Library than the limit specifies, then **we do respect the sub rule**, if we do α and β[377].

[374] cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp., §3; emphasis and additional braces KR..
[375] follows by Modus Ponens to $(\!(\Lambda \to \Xi)\!)$.
[376] follows by Modus Ponens to $(\!(\Lambda \to \Xi)\!)$.
[377] follows by Modus Ponens to $(\!(\omega \to (\alpha \wedge \beta))\!)$.

- If the object code contains elements of the Library at most up to specified limits, then **we do respect the sub rule** without having to do some additionally tasks[378]
- But if the object code contains more and/or larger elements of the Library than the limit specifies **and** if we do not do α *or* β, then **we do not respect the sub rule**[379].

Thus, – at the end and based on the additional object code specification and the known empirical background knowledge concerning the software programming – the LGPL3-RevEng-Rule delivers the same result as the LGPL2-RevEng-Rule[380]:

- *With respect to a LGPL-v3 licensed Library, you are not required to allow reverse engineering, if you [A] develop your work using the Library, on the base of a standard version of the Library containing the interfaces as the original developers have designed it, if you [B] compile your work using this Library, as a discrete (set of) dynamically linkable or combinable file(s), if you [C] use only the standard compilation methods which preserve the upstream approved interfaces*[381]*, and if you [D] distribute the produced unlinked object code or bytecode files before they are linked as an executable.*
- *In all other cases of distributing a work using such a Library, you are required to allow reverse engineering of the work using this Library – especially, ...*
  - *if you distribute the work using the Library and the Library together as a statically linked program or as an integrated package containing both parts, the work using the library and the Library itself*[382]*.*
  - *if you distribute a work containing manually copied portions of the Library.*

3.4.3 Reverse Engineering in the other Open Source Licenses

The rest of our way is simple: First, we can ascertain, that none of the other open source licenses we consider[383], contain the phrase 'reverse engineering'. Moreover,

[378] follows by definition of an implication: if the premise of this sub rule is false, the sub rule is as whole is true and hence respected.
[379] follows from definition of an implication: if the premise is true and the conclusion is false, the the implication as whole is false, as well.
[380] → 90
[381] and which therefore do not to exceed the LGPL-v3 limits
[382] This holds also if you distribute a script language based program or package, notwithstanding the fact, that one does not need the permission of reverse engineering to understand script language based applications
[383] → p. 71

they even do not contain one of the single words[384]. So, we may infer, that these most important other open source licenses could at most indirectly require the permission of reverse engineering. Second, we know already that distributing script code let the allowance to reverse engineer, become irrelevant: script code can directly be read and understood, if one knows the script language[385]. Third, from the definition of strong copleft we may derive, that distributing software licensed under a strong copyleft license let the permission of reverse engineering become unimportant, because the source code of the work using the libraries licensed under a copleft license, must also be made accessible[386].

So – overally – we may conclude, that we have only to consider those cases, where a piece of software is distributed in form of binaries or bytecode, which uses libraries licensed under permissive open source licenses or under weak copyleft licenses.

From the definition of being a permissive license or a weak copyleft license we know already that the licenses of the open source components do not directly influence the permission or interdiction to use the overarching work which uses the open source software components[387].

So, if we distribute such a work in form of dynamically linkable, but still not linked binaries or bytecode files, then there is no way to reasonably derive that the work using the components, may be reverse engineered: The permissive or weak copyleft open source licenses mainly concern the open source components, not the work using the components. On the one side, these licenses indeed require that we add the license texts and the copyright lines of all the open source components our work wants to use, to the distributed package containing our work. And the lisenses prohibit to modify the licensing assertions being integrated into the open source components our work wants to use[388]. But – on the other side and in accordance to the permissive or weak-copyleft licenses – the freedom to use, to study, to modify, or to distribute the software, which is established by these open source licenses, concerns only the open source components themselves, not the work using the open source components. So, as long as these components still are not linked to or combined with the using work in accordance to the standard

[384] One can verify this negative statement by (a) loading down all licenses from the OSI homepage (http://opensource.org/licenses/alphabetical) and by (b) executing the command `grep -i "engineering" *` respectively `grep -i "reverse" *` in the directory into which the license files have been stored: grep will find the words *reverse* and *engineering* only in the texts of the LGPLs.

[385] → p. 85

[386] cf. *Stallman*: What is Copyleft?, 1996, wp.

[387] cf. *Reincke, Karsten*, *Greg Sharpe*, a. *contributors*: Open Source License Compendium. How to Achieve Open Source License Compliance; 2015 ⟨URL: `http://www.oslic.org/releases/oslic.pdf`⟩ – reference download: 2015-01-20, pp. 20ff..

[388] These requirements are part of all the open source licenses we consider here. For details cf. id., l.c., pp. chapter 6.

compilation and computation methods, they can indeed be studied or modified without the need to study or modify the work which uses these components[389].

On the other side, if we compliantly distribute the work using the components, as a statically linked binary or bytecode file – which therefore already contains all the necessary components[391] and can directly be executed –, then we are also obliged to add all the open source license texts and all the copyright lines to our package, and we are not allowed to modify one of the licensing assertions integrated into the original open source components[392]. Thus, one might conclude, that the freedom to use and to modify the open source components themselves, survive if we distribute software statically linked to or combined with the open source components. So, the receiver of the statically linked work probably is allowed to modify the embedded open source components - even if he had to edit the binary or bytecode files. Methods to develop binary files reversely, are known as reverse engineering. Hence, if we distribute a statically linked work using open source licensed components, we have at least to fear that our receivers indirectly have also got the permission to reverse engineer our complete product. And we have to fear so even if the statically linked libraries are licensed under any permissive or weak copleft license.

So, again, we can summarize the result in the following form:

- *With respect to a Library licensed under any permissive or weak copyleft license, you are not required to allow reverse engineering, if you [A] develop your work using the Library, on the base of a standard version of the Library containing the interfaces as the original developers have designed it, if you [B] compile your work using this Library, as a discrete (set of) dynamically linkable or combinable file(s), if you [C] use only the standard compilation*

389) The only way to infer that the licenses of the components operates also on the using work, is to argue that the using work must at least contain elements (identifiers etc.) of the interfaces declared (but not defined) by the libraries and that therefore at least these elements may be investigated or modified. This challenge is explicitly addressed by the LGPL[390]. Fortunately, it is a general feature of software libraries that they must and shall be used in accordance to the interfaces, the developers of the libraries have designed to make their libraries practically usable. So, if the licenses – in contrary to the LGPLs – do not explicitly address the issue of implicitly included portions of the library in case of unlinked binaries or bytecode files which have been compiled in accordance to the standard methods and which therefore use open source software by reffering to their standard interfaces, then one has to infer from the nature of computation, that the developers have implictly allowed without any requirements such an integration of declared, but not defined interface elements, because they have designed the interface as they did and because they have licensed their work as they did. If they had not wished to use these elements without any requirements, hey had designed another interface. And if they had wished to incorporate any copyleft effect or permission of reverse engineering, then they would have selected another license. But again: this conclusion holds only for the standard methods to use a software library.

391) instead of only the declared interface elements!

392) cf. *Reincke, Sharpe, a. other contributors*: OSLiC, 2015, pp. chapter 6..

methods which preserve the upstream approved interfaces, and if you [D] distribute the produced unlinked object code or bytecode files before they are linked as an executable.

- *In all other cases of distributing a work using such a Library, you have at least to fear that you are implictly allowing reverse engineering of the work using this Library – especially, ...*
  - *if you distribute the work using the Library and the Library together as a statically linked program or as an integrated package containing both parts, the work using the library and the Library itself*[393].
  - *if you distribute a work containing manually copied portions of the Library.*

3.4.4 Reverse Engineering in Open Source Licenses: Summary

So, finally we can compile all our results into one single result:

- *With respect to any open source Library*[394]*, you are not required to allow reverse engineering, if you [A] develop your work using the Library, on the base of a standard version of the Library containing the interfaces as the original developers have designed it, if you [B] compile your work using this Library, as a discrete (set of) dynamically linkable or combinable file(s), if you [C] use only the standard compilation methods which preserve the upstream approved interfaces*[395]*, and if you [D] distribute the produced unlinked object code or bytecode files before they are linked as an executable.*
- *In all other cases of distributing your work using such a Library, you are probably required to allow reverse engineering of your work. By all means, you have at least to fear that you are implictly allowing reverse engineering of your work using such a Library – especially, ...*
  - *if you distribute the work using the Library and the Library together as a statically linked program or as an integrated package containing both parts, the work using the library and the Library itself*[396].
  - *if you distribute a work containing manually copied portions of the Library.*

393) This holds also if you distribute a script language based program or package, notwithstanding the fact, that one does not need the permission of reverse engineering to understand script language based applications

394) → p. 71

395) and which therefore do not to exceed limits, prescribed by the owners of the Library

396) This holds also if you distribute a script language based program or package, notwithstanding the fact, that one does not need the permission of reverse engineering to understand script language based applications

And, so, we can reformulate our result as a slightly modified "rule of thumbs" originally offered by an open source expert who analyzed the problem of protecting your own work from an other viewport:

- "DO NOT statically link [or combine] [open source] code if you wish to keep your program proprietary [and if you want to protect it against reverse engineering]"[397].
- "DO dynamically link to [any open source code, not only to] LGPL code"[398].

q.e.d

3.5 Excursion: The problem of license compatibility [tbd]

Here we discuss the often neglected or only superficially treated 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 will argue that the question can simply be answered: Only if you embed two libraries which both are licensed under an on-top-development protecting license and if both these 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.

...

3.6 Excursion: open source software and money [tbd]

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

...

397) cf. *Ilardi*: Common OSS License Problems, 2010, pp. 6; bracketed text KR..
398) cf. id., ibid.

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 first 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 second in the same context that an open source license "[...] shall not require a royalty or other fee for such sale"[399].

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 context[400].

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

[399] cf. *Open Source Initiative*: The Open Source Definition, 2012, wp §1.

[400] Fortunately, sometimes one task list fulfills the conditions of more than one use case—a welcome reduction of complexity

the 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 sense[401]. 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 you might intend to hand over the software (also) to other people. More

[401] 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 ;-)

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 '2others'.

- The **form of the distributed files**: Many licenses also draw a distinction between distributing the software as sources and distributing the files as binaries. In this case, we will ask you, whether you want to distribute the software in the form of binaries or as source code. In the first case, the answer should be 'binaries', in the second 'sources'
- The kind of the **ioAccess of the executed program**: At least one license draws a distinction between an open source based work offering only local access to its io data and an open source based work distributing its io data via internet. In the first case, the answer should be 'onlyLocally', in the second 'viaInternet'

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::2others`
- `form::binaries` *or* `form::sources`
- `ioAccess::onlyLocally` *or* `ioAccess::viaInternet`

We have already defined the *open source use case* as the combination of these tokens. If we simply combine all these tokens of all these classes with all the tokens of the other classes[402], we get $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 62$ sets of tokens—or 62 *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. 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 open source software is a *snimoli*—a snippet of code, a module, a plugin,

[402] in the sense of the cross product TYPE × STATE × CONTEXT × RECIPIENT × FORM × IOACCESS. In some earlier versions of the OSLiC, we also asked 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. Meanwhile, we clearly discovered that it is unnecessary to increase the complexity by the results of this question. For Details → OSLiC p. 61

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.

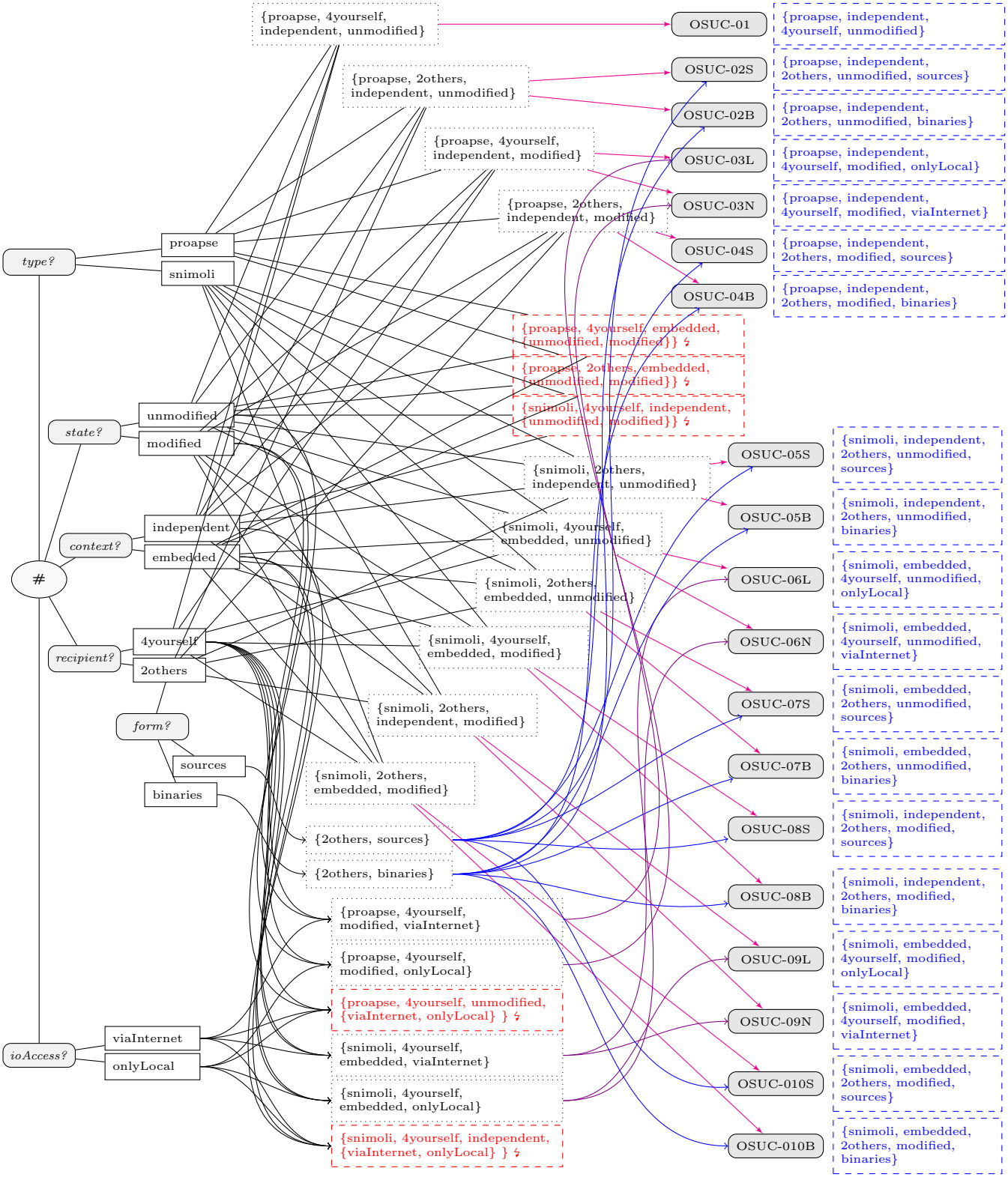
2. If you already have specified that the used open source software is a *snimoli*—a snippet of code, a module, a plugin, or a library—and that this *snimoli* shall be used only by yourself (not distributed to other 3rd. parties) then your answer must also imply that this *snimoli* is used in combination, 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.
3. To enquire the *form* of the distributed files is only relevant if you have decided to distribute the software to other recipients *2others*.
4. With respect to the one license using the type of ioAccess as a discriminator, it is only relevant to enquire the type of the ioAccess if you either locally execute a *modified* open source program *4yourself* or if you locally execute a program *4yourself*, which uses an *embedded* open source component, regardless whether it has been modified or not.

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 diagram summarizes the main results of our investigation[403]::

[403] Each of the invalid use cases (= sets of tokens) [for details s. p. 105] is marked by an ↯ and leads to an empty set (= ∅). We are using the word 'invalid' a little ambigiuosly: A combination of values is invalid, if it is empirically impossible, to combine the features or if it is irrelavant to subclassify a concept by the added features. Particularly:

- A proapse can not be embedded into another software unit, also containing a main-function.
- Using a software library only for yourself and independently (not in combination with larger software unit), is like having an unused heap of bytes on your disc.
- To discriminate between sources and binaries is only valid in case of distributing software.
- To discriminate between an executed program with an only locally based io access and that with an internet based io access is only relevant, if you are using the software for yourself what implies to execute it.

.


{proapse, 4yourself, independent, unmodified}
{proapse, 2others, independent, unmodified}
{proapse, 4yourself, independent, modified}
{proapse, 2others, independent, modified}
{proapse, 4yourself, embedded, {unmodified, modified}} ↯
{proapse, 2others, embedded, {unmodified, modified}} ↯
{snimoli, 4yourself, independent, {unmodified, modified}} ↯
{snimoli, 2others, independent, unmodified}
{snimoli, 4yourself, embedded, unmodified}
{snimoli, 2others, embedded, unmodified}
{snimoli, 4yourself, embedded, modified}
{snimoli, 2others, independent, modified}
{snimoli, 2others, embedded, modified}
{2others, sources}
{2others, binaries}
{proapse, 4yourself, modified, viaInternet}
{proapse, 4yourself, modified, onlyLocal}
{proapse, 4yourself, unmodified, {viaInternet, onlyLocal} } ↯
{snimoli, 4yourself, embedded, viaInternet}
{snimoli, 4yourself, embedded, onlyLocal}
{snimoli, 4yourself, independent, {viaInternet, onlyLocal} } ↯
type?
proapse
snimoli
state?
unmodified
modified
context?
independent
embedded
#
recipient?
4yourself
2others
form?
sources
binaries
ioAccess?
viaInternet
onlyLocal
OSUC-01
OSUC-02S
OSUC-02B
OSUC-03L
OSUC-03N
OSUC-04S
OSUC-04B
OSUC-05S
OSUC-05B
OSUC-06L
OSUC-06N
OSUC-07S
OSUC-07B
OSUC-08S
OSUC-08B
OSUC-09L
OSUC-09N
OSUC-010S
OSUC-010B
{proapse, independent, 4yourself, unmodified}
{proapse, independent, 2others, unmodified, sources}
{proapse, independent, 2others, unmodified, binaries}
{proapse, independent, 4yourself, modified, onlyLocal}
{proapse, independent, 4yourself, modified, viaInternet}
{proapse, independent, 2others, modified, sources}
{proapse, independent, 2others, modified, binaries}
{snimoli, independent, 2others, unmodified, sources}
{snimoli, independent, 2others, unmodified, binaries}
{snimoli, embedded, 4yourself, unmodified, onlyLocal}
{snimoli, embedded, 4yourself, unmodified, viaInternet}
{snimoli, embedded, 2others, unmodified, sources}
{snimoli, embedded, 2others, unmodified, binaries}
{snimoli, independent, 2others, modified, sources}
{snimoli, independent, 2others, modified, binaries}
{snimoli, embedded, 4yourself, modified, onlyLocal}
{snimoli, embedded, 4yourself, modified, viaInternet}
{snimoli, embedded, 2others, modified, sources}
{snimoli, embedded, 2others, modified, binaries}

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

This chapter offers the Open Source Use Case Finder*: Based on the information gathered by a form, it allows to traverse a tree whose leaves are linked to the* open source use cases *which finally refer to the respective to-do lists.*

5.1 A standard form for gathering the relevant information

	Which open source software do you want to use?	
	Under which open source license is it released?	
Focus	**Questions**	**Answers**
Type	*Is the open source software you want to use a library in the broadest sense (an includable code **sni**ppet, a linkable **mo**dule or **li**brary, or a loadable plugin), or is it an autonomous **pro**gram, **ap**plication, or **se**rver which can be executed?*	☐ snimoli ☐ proapse
State	*Do you want to leave the open source software **unmodified** as you have received it, or are you going to create a **modified** version of it?*	☐ unmodified ☐ modified
Context	*Are you going to use / distribute the open source software as an **independent** unit, or do you plan to integrate it as an **embedded** component into a complexer piece of software?*	☐ independent ☐ embedded
Recipient	*Are you going to use the open source software only **for yourself**, or do you plan to (re)distribute it (also) **to other** third parties?*	☐ 4yourself ☐ 2others
Form	*Given you want to (re)distribute an open source based work [2others], do you focus on distributing the **binaries** or the **sources**?*	☐ binaries ☐ sources
IoAccess	*Given you are using open source software [4yourself] by executing a modified os program [modified] or by creating & executing a program using an os library [embedded], does this program distribute its IO data **only locally** or **via internet**?*	☐ onlyLocally ☐ viaInternet

As discussed earlier, there are of course some invalid or irrelevant combinations.[404] Here are some extra explanations concerning the classes resp. the focuses:

Type: A piece of (open source) software is a program, an application, or a server, only 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 which takes the file as one of its arguments and executes the commands.

State: You are modifying a piece of (open source) software if you expand, reduce or modify at 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. But if you only modify some of the configuration files, you are not modifying the open source software itself.

Context: You are using a piece of open source software as an embedded component of a larger unit . . .

- if one of your files of the larger unit contains a verbatim or a modified copy (i.e. a snippet) of the received open source software, or
- if your larger unit contains an include statement referring to a functionally defining file of the received open source software, or
- if your larger unit calls a function defined in the received open source software, or
- if your development environment contains a compiler or linker directive referring to the received open source software (binaries) and if your larger unit can't be executed without resolving this linker directive.

Recipient: You are using 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 hand the open source software over to third parties.[405]

[404] type::proapse excludes state::embedded; recipient::4yourself excludes the combination with state::independent and type::snimoli; any value of class 'mode' implies state::embedded; form is only relevant if recipient::2others; ioAccess is only relevant if recipient::4yourself[for details see page 105]. If you have encountered one of these invalid combinations, please check the corresponding explanations.

[405] 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.

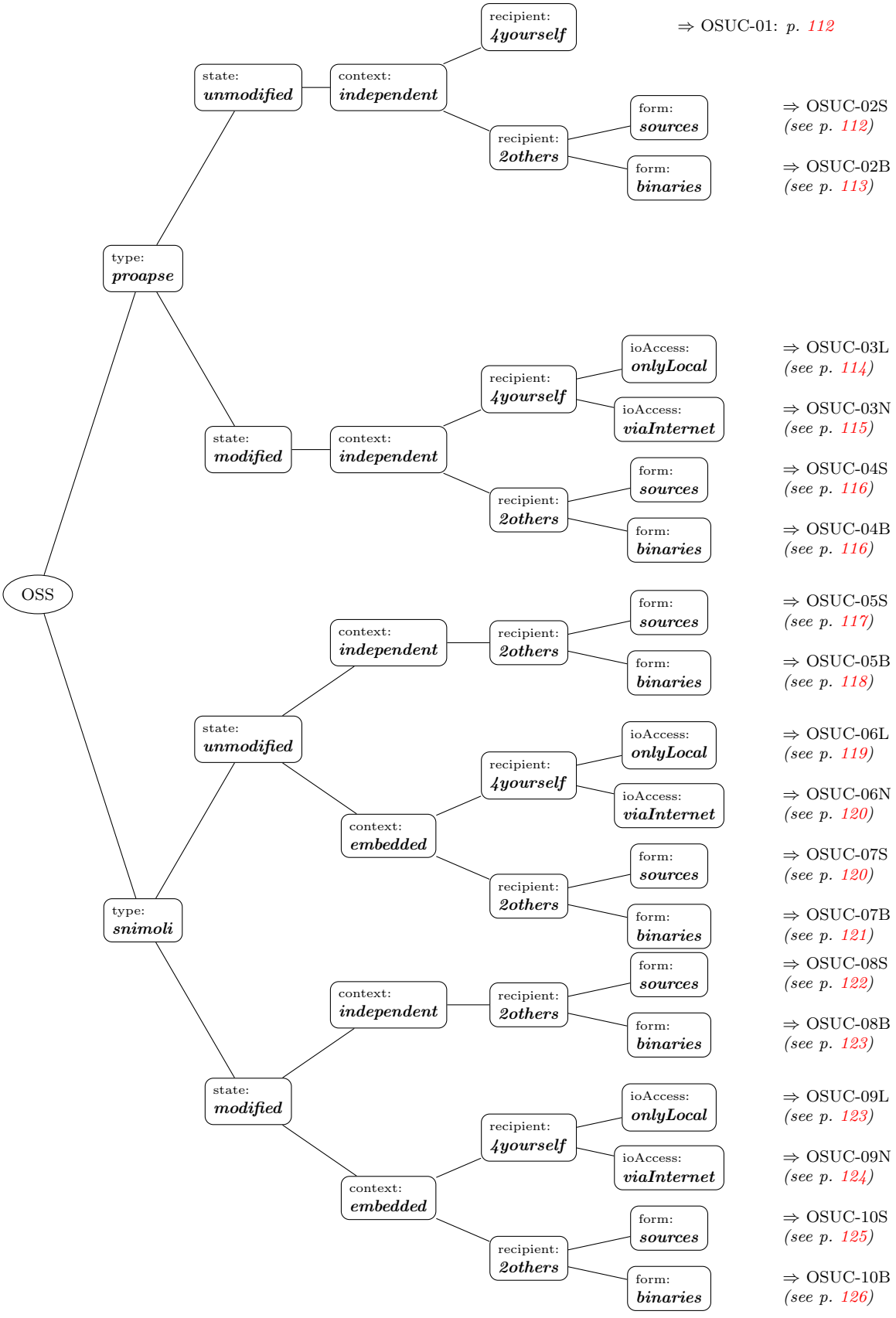
Form: Open source software knows two ways to distribute the software: in the form of binaries and in the form of sources. Mostly it is up to you to decide whether you want to distribute only the binaries or whether you are intentionally going to distribute the sources (too). At a first glance, the concepts 'sources' and 'binaries' seems to be clearly distinguished. On the one hand, compiled sources should be taken as binaries. On the other hand, editable pieces of software are denoted by the concept 'sources'. But sometimes the difference is not as clear as wished: For example, you can modify even already compiled object files by using an hex-editor. Or it is very difficult to modify the minimized versions of javascript files even if they are indeed text files. Therefore, the OSLiC 'reuses' a famous **rule of thumb**: "The source code for a work means the preferred form of the work for making modifications to it".[406] All other forms are denoted by the concept of 'binaries'. Based on this specification, you can respect some special conditions if you want to distribute the sources and/or the binaries.

ioAccess: If you execute an open source program or an own program using an open source library, then (normally) you do not distribute that software. Under these circumstances, the most open source licenses do not require anything for executing the program compliantly - even if it is the base of a globally used internet service. For closing this 'gap', the AGPL has been invented: Like the GPL, the AGPL let the obligation to fulfill the well known set of GPL tasks be triggered by distributing the software. But, it let these tasks also be triggered by an established remote network interaction: whoever interacts with the locally executed program remotely through a computer network gets all the rights which normally the receiver of a distribution gets. Nevertheless, the AGPL does not wish to cause an overhead of tasks: Only *locally excuted open source programs which have been modfied* or *locally executed own programs using an AGPL licensed library* shall indeed trigger the fulfillment of the requirements. Thus, we introduced the features *ioAccess:onlyLocally* and *ioAccess:viaInternet*: They are only relevant if you uses a program only for yourself (4yourself) **and** [(if that AGPL licensed program has been modified {proapse and modified}) **or** (if that program uses an embedded AGPL licensed library {snimoli and embedded})].

5.2 The taxonomic Open Source Use Case Finder

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

406) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp §3.


OSS
type: proapse
state: unmodified
context: independent
recipient: 4yourself
⇒ OSUC-01: p. 112
recipient: 2others
form: sources
⇒ OSUC-02S (see p. 112)
form: binaries
⇒ OSUC-02B (see p. 113)
state: modified
context: independent
recipient: 4yourself
ioAccess: onlyLocal
⇒ OSUC-03L (see p. 114)
ioAccess: viaInternet
⇒ OSUC-03N (see p. 115)
recipient: 2others
form: sources
⇒ OSUC-04S (see p. 116)
form: binaries
⇒ OSUC-04B (see p. 116)
type: snimoli
state: unmodified
context: independent
recipient: 2others
form: sources
⇒ OSUC-05S (see p. 117)
form: binaries
⇒ OSUC-05B (see p. 118)
context: embedded
recipient: 4yourself
ioAccess: onlyLocal
⇒ OSUC-06L (see p. 119)
ioAccess: viaInternet
⇒ OSUC-06N (see p. 120)
recipient: 2others
form: sources
⇒ OSUC-07S (see p. 120)
form: binaries
⇒ OSUC-07B (see p. 121)
state: modified
context: independent
recipient: 2others
form: sources
⇒ OSUC-08S (see p. 122)
form: binaries
⇒ OSUC-08B (see p. 123)
context: embedded
recipient: 4yourself
ioAccess: onlyLocal
⇒ OSUC-09L (see p. 123)
ioAccess: viaInternet
⇒ OSUC-09N (see p. 124)
recipient: 2others
form: sources
⇒ OSUC-10S (see p. 125)
form: binaries
⇒ OSUC-10B (see p. 126)

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

On the following pages, each **O**pen **S**ource **U**se **C**ase is textually specified one more time and complemented by a list of page numbers. Each of these pages covers the license-specific to-do list whose items together offer a processable way for acting according to the license under the circumstances of the described **O**pen **S**ource **U**se **C**ase.

OSUC-01: Only for yourself, you are going to use an unmodified open source program, application, or server just as you received it. But you do not 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. 128 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 150 for the **Apache-2.0** *(= Apache License)*
- p. 168 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 161 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 176 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 181 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 196 for the **EUPL-1.1** *(= European Union Public License)*
- p. 211 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 224 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 243 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 254 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 272 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 277 for the **MPL** *(= Mozilla Public License)*
- p. 291 for the **MS-PL** *(= Microsoft Public License)*
- p. 298 for the **PostgreSQL** *(= Postgres License)*
- p. 302 for the **PHP-3.0** License

OSUC-02S: Just as you received it, you are going to distribute an unmodified open source program, application, or server to third parties in the form

of sources. 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, 2others, sources*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 129 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 150 for the **Apache-2.0** *(= Apache License)*
- p. 168 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 162 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 177 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 182 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 197 for the **EUPL-1.1** *(= European Union Public License)*
- p. 212 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 224 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 244 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 255 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 273 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 278 for the **MPL** *(= Mozilla Public License)*
- p. 292 for the **MS-PL** *(= Microsoft Public License)*
- p. 299 for the **PostgreSQL** *(= Postgres License)*
- p. 303 for the **PHP-3.0** License

OSUC-02B: Just as you received it, you are going to distribute an unmodified open source program, application, or server to third parties in the form of binaries. 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, 2others, binaries*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 130 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 151 for the **Apache-2.0** *(= Apache License)*
- p. 169 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*

- p. 162 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 177 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 183 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 197 for the **EUPL-1.1** *(= European Union Public License)*
- p. 212 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 225 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 244 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 256 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 273 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 279 for the **MPL** *(= Mozilla Public License)*
- p. 292 for the **MS-PL** *(= Microsoft Public License)*
- p. 299 for the **PostgreSQL** *(= Postgres License)*
- p. 304 for the **PHP-3.0** License

OSUC-03L: You are executing an open source program, application, or server which you have modified (but not combined with other components in the sense of software development) and which distributes its input/output only locally to you (= *proapse, modified, independent, 4yourself, onlyLocal*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 128 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 150 for the **Apache-2.0** *(= Apache License)*
- p. 168 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 161 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 176 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 181 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 196 for the **EUPL-1.1** *(= European Union Public License)*
- p. 211 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 224 for the **GPL-3.0** *(= GNU General Public License Version 3)*

- p. 243 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 254 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 272 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 277 for the **MPL** *(= Mozilla Public License)*
- p. 291 for the **MS-PL** *(= Microsoft Public License)*
- p. 298 for the **PostgreSQL** *(= Postgres License)*
- p. 302 for the **PHP-3.0** License

OSUC-03N: You are executing an open source program, application, or server which you have modified (but not combined with other components in the sense of software development) and which distributes its input/output to you or other users via the internet (= *proapse, modified, independent, 4yourself, viaInternet*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 141 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 150 for the **Apache-2.0** *(= Apache License)*
- p. 168 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 161 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 176 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 181 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 196 for the **EUPL-1.1** *(= European Union Public License)*
- p. 211 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 224 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 243 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 254 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 272 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 277 for the **MPL** *(= Mozilla Public License)*
- p. 291 for the **MS-PL** *(= Microsoft Public License)*

- p. 298 for the **PostgreSQL** *(= Postgres License)*
- p. 302 for the **PHP-3.0** License

OSUC-04S: You are going to modify an open source program, application, or server after you received it and before you will distribute it to third parties in the form of sources. But you do not combine this modified program, application, or server with other software components in the sense of software development (= *proapse, modified, independent, 2others, sources*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 133 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 152 for the **Apache-2.0** *(= Apache License)*
- p. 169 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 163 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 177 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 184 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 201 for the **EUPL-1.1** *(= European Union Public License)*
- p. 216 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 228 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 248 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 259 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 273 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 280 for the **MPL** *(= Mozilla Public License)*
- p. 292 for the **MS-PL** *(= Microsoft Public License)*
- p. 299 for the **PostgreSQL** *(= Postgres License)*
- p. 304 for the **PHP-3.0** License

OSUC-04B: You are going to modify an open source program, application, or server after you received it and before you will distribute it to third parties in the form of binaries. But you do not combine this modified program, application, or server with other software components in the sense of software development (= *proapse, modified, independent, 2others, binaries*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 134 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 153 for the **Apache-2.0** *(= Apache License)*
- p. 170 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 164 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 178 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 185 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 202 for the **EUPL-1.1** *(= European Union Public License)*
- p. 217 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 230 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 248 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 260 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 273 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 281 for the **MPL** *(= Mozilla Public License)*
- p. 293 for the **MS-PL** *(= Microsoft Public License)*
- p. 299 for the **PostgreSQL** *(= Postgres License)*
- p. 305 for the **PHP-3.0** License

OSUC-05S: Just as you received it, you are going to distribute an unmodified open source library, code snippet, module, or plugin to third parties in the form of sources. 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, 2others, sources*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 129 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 150 for the **Apache-2.0** *(= Apache License)*
- p. 168 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 162 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*

- p. 177 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 182 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 197 for the **EUPL-1.1** *(= European Union Public License)*
- p. 212 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 224 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 244 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 255 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 273 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 278 for the **MPL** *(= Mozilla Public License)*
- p. 292 for the **MS-PL** *(= Microsoft Public License)*
- p. 299 for the **PostgreSQL** *(= Postgres License)*
- p. 303 for the **PHP-3.0** License

OSUC-05B: Just as you received it, you are going to distribute an unmodified open source library, code snippet, module, or plugin to third parties in the form of binaries. 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, 2others, binaries*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 130 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 151 for the **Apache-2.0** *(= Apache License)*
- p. 169 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 162 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 177 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 183 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 197 for the **EUPL-1.1** *(= European Union Public License)*
- p. 212 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 225 for the **GPL-3.0** *(= GNU General Public License Version 3)*

- p. 244 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 256 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 273 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 279 for the **MPL** *(= Mozilla Public License)*
- p. 292 for the **MS-PL** *(= Microsoft Public License)*
- p. 299 for the **PostgreSQL** *(= Postgres License)*
- p. 304 for the **PHP-3.0** License

OSUC-06L: You are executing any application which distributes input/output only locally to you and which uses an unmodified embedded open source library, code snippet, module, or plugin (= *snimoli, umodified, embedded, 4yourself, onlyLocal*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 128 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 150 for the **Apache-2.0** *(= Apache License)*
- p. 168 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 161 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 176 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 181 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 196 for the **EUPL-1.1** *(= European Union Public License)*
- p. 211 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 224 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 243 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 254 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 272 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 277 for the **MPL** *(= Mozilla Public License)*
- p. 291 for the **MS-PL** *(= Microsoft Public License)*
- p. 298 for the **PostgreSQL** *(= Postgres License)*

- p. 302 for the **PHP-3.0** License

OSUC-06N: You are executing any application which distributes its input/output to you or other users via the internet and which uses an unmodified embedded open source library, code snippet, module, or plugin (= *snimoli, umodified, embedded, 4yourself, viaInternet*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 143 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 150 for the **Apache-2.0** *(= Apache License)*
- p. 168 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 161 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 176 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 181 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 196 for the **EUPL-1.1** *(= European Union Public License)*
- p. 211 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 224 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 243 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 254 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 272 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 277 for the **MPL** *(= Mozilla Public License)*
- p. 291 for the **MS-PL** *(= Microsoft Public License)*
- p. 298 for the **PostgreSQL** *(= Postgres License)*
- p. 302 for the **PHP-3.0** License

OSUC-07S: Just as you received it and before you will distribute it to third parties in the form of sources and together with a larger software unit, you are going to combine and embed an unmodified open source library, code snippet, module, or plugin into that larger software unit in the sense of software development (= *snimoli, unmodified, embedded, 2others, sources*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 131 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 150 for the **Apache-2.0** *(= Apache License)*

- p. 168 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 162 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 177 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 182 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 199 for the **EUPL-1.1** *(= European Union Public License)*
- p. 213 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 226 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 246 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 257 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 273 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 278 for the **MPL** *(= Mozilla Public License)*
- p. 292 for the **MS-PL** *(= Microsoft Public License)*
- p. 299 for the **PostgreSQL** *(= Postgres License)*
- p. 303 for the **PHP-3.0** License

OSUC-07B: Just as you received it and before you will distribute it to third parties in the form of binaries and together with a larger software unit, you are going to combine and embed an unmodified open source library, code snippet, module, or plugin into that larger software unit in the sense of software development (= *snimoli, unmodified, embedded, 2others, binaries*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 132 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 151 for the **Apache-2.0** *(= Apache License)*
- p. 169 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 162 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 177 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 183 for the **EPL-1.0** *(= Eclipse Public License)*

- p. 199 for the **EUPL-1.1** *(= European Union Public License)*
- p. 214 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 227 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 246 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 258 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 273 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 279 for the **MPL** *(= Mozilla Public License)*
- p. 292 for the **MS-PL** *(= Microsoft Public License)*
- p. 299 for the **PostgreSQL** *(= Postgres License)*
- p. 304 for the **PHP-3.0** License

OSUC-08S: Before you will distribute it to third parties in the form of sources, you are going to modify an open source library, code snippet, module, or plugin. But you do not combine it with other software components in the sense of software development (= *snimoli, modified, independent, 2others, sources*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 136 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 154 for the **Apache-2.0** *(= Apache License)*
- p. 171 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 164 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 178 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 186 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 203 for the **EUPL-1.1** *(= European Union Public License)*
- p. 218 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 231 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 249 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 262 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*

- p. 274 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 283 for the **MPL** *(= Mozilla Public License)*
- p. 294 for the **MS-PL** *(= Microsoft Public License)*
- p. 300 for the **PostgreSQL** *(= Postgres License)*
- p. 306 for the **PHP-3.0** License

OSUC-08B: Before you will distribute it to third parties in the form of binaries, you are going to modify an open source library, code snippet, module, or plugin. But you do not combine it with other software components in the sense of software development (= *snimoli, modified, independent, 2others, binaries*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 137 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 155 for the **Apache-2.0** *(= Apache License)*
- p. 171 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 165 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 178 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 188 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 204 for the **EUPL-1.1** *(= European Union Public License)*
- p. 220 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 232 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 250 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 263 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 274 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 284 for the **MPL** *(= Mozilla Public License)*
- p. 295 for the **MS-PL** *(= Microsoft Public License)*
- p. 300 for the **PostgreSQL** *(= Postgres License)*
- p. 307 for the **PHP-3.0** License

OSUC-09L: You are executing any application which distributes input/output only locally to you and which uses an embedded open source library, code

snippet, module, or plugin – being modified by you (= *snimoli, modified, embedded, 4yourself, onlyLocal*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 128 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 150 for the **Apache-2.0** *(= Apache License)*
- p. 168 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 161 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 176 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 181 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 196 for the **EUPL-1.1** *(= European Union Public License)*
- p. 211 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 224 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 243 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 254 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 272 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 277 for the **MPL** *(= Mozilla Public License)*
- p. 291 for the **MS-PL** *(= Microsoft Public License)*
- p. 298 for the **PostgreSQL** *(= Postgres License)*
- p. 302 for the **PHP-3.0** License

OSUC-09N: You are executing any application which distributes its input/output to you or other users via the internet and which uses an embedded open source library, code snippet, module, or plugin – being modified by you (= *snimoli, modified, embedded, 4yourself, viaInternet*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 143 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 150 for the **Apache-2.0** *(= Apache License)*
- p. 168 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 161 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*

- p. 176 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 181 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 196 for the **EUPL-1.1** *(= European Union Public License)*
- p. 211 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 224 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 243 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 254 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 272 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 277 for the **MPL** *(= Mozilla Public License)*
- p. 291 for the **MS-PL** *(= Microsoft Public License)*
- p. 298 for the **PostgreSQL** *(= Postgres License)*
- p. 302 for the **PHP-3.0** License

OSUC-10S: Before you will distribute it to third parties in the form of sources, you are going to modify an open source library, code snippet, module, or plugin, which you combine with other software components in the sense of software development (= *snimoli, modified, embedded, 2others, sources*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 138 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 156 for the **Apache-2.0** *(= Apache License)*
- p. 172 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 166 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 179 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 189 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 205 for the **EUPL-1.1** *(= European Union Public License)*
- p. 221 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 234 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 251 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*

- p. 264 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 274 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 285 for the **MPL** *(= Mozilla Public License)*
- p. 295 for the **MS-PL** *(= Microsoft Public License)*
- p. 300 for the **PostgreSQL** *(= Postgres License)*
- p. 307 for the **PHP-3.0** License

OSUC-10B: Before you will distribute it to third parties in the form of binaries, you are going to modify an open source library, code snippet, module, or plugin, which you combine with other software components in the sense of software development (= *snimoli, modified, embedded, 2others, binaries*). To see the *specific, license fulfilling to-do lists* jump to the following pages:

- p. 140 for the **AGPL-3.0** *(= GNU Affero General Public License)*
- p. 157 for the **Apache-2.0** *(= Apache License)*
- p. 173 for the **BSD-2-Clause** License *(= Berkeley Software Distribution)*
- p. 167 for the **BSD-3-Clause** License *(= Berkeley Software Distribution)*
- p. 179 for the **CDDL-1.0** *(= Common Develop and Distribution License)*
- p. 190 for the **EPL-1.0** *(= Eclipse Public License)*
- p. 207 for the **EUPL-1.1** *(= European Union Public License)*
- p. 222 for the **GPL-2.0** *(= GNU General Public License Version 2)*
- p. 235 for the **GPL-3.0** *(= GNU General Public License Version 3)*
- p. 253 for the **LGPL-2.1** *(= GNU Lesser General Public License Version 2.1)*
- p. 265 for the **LGPL-3.0** *(= GNU Lesser General Public License Version 3)*
- p. 274 for the **MIT** License *(= Massachusetts Institute of Technology)*
- p. 287 for the **MPL** *(= Mozilla Public License)*
- p. 296 for the **MS-PL** *(= Microsoft Public License)*
- p. 300 for the **PostgreSQL** *(= Postgres License)*
- p. 309 for the **PHP-3.0** License

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 points which are relevant for many of the to-do lists. So that the same points are not repeated too often, we will start with these general remarks and refer to them throughout the chapter.

- Sometimes when delivering a binary package containing open source software, the medium doesn't allow the recipient to view all files contained in that package. For example, a lot of mobile devices don't give the user access to the file system. But open source licenses often require 'to give' someone copies of text files, such as the license text, copyright notes, or specific notice file. The safe interpretation of 'giving someone a text' is that the receiver must be able to read it[407]. Thus, 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 through the UI of your application—for example in a specific copyright screen[408]. The OSLiC does not want to refine the taxonomies down to the level of operating systems, so it is up to the user to keep this in mind when reading 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 they will offer the files all the time your product is being distributed[409]. Even if it would be safe to assume

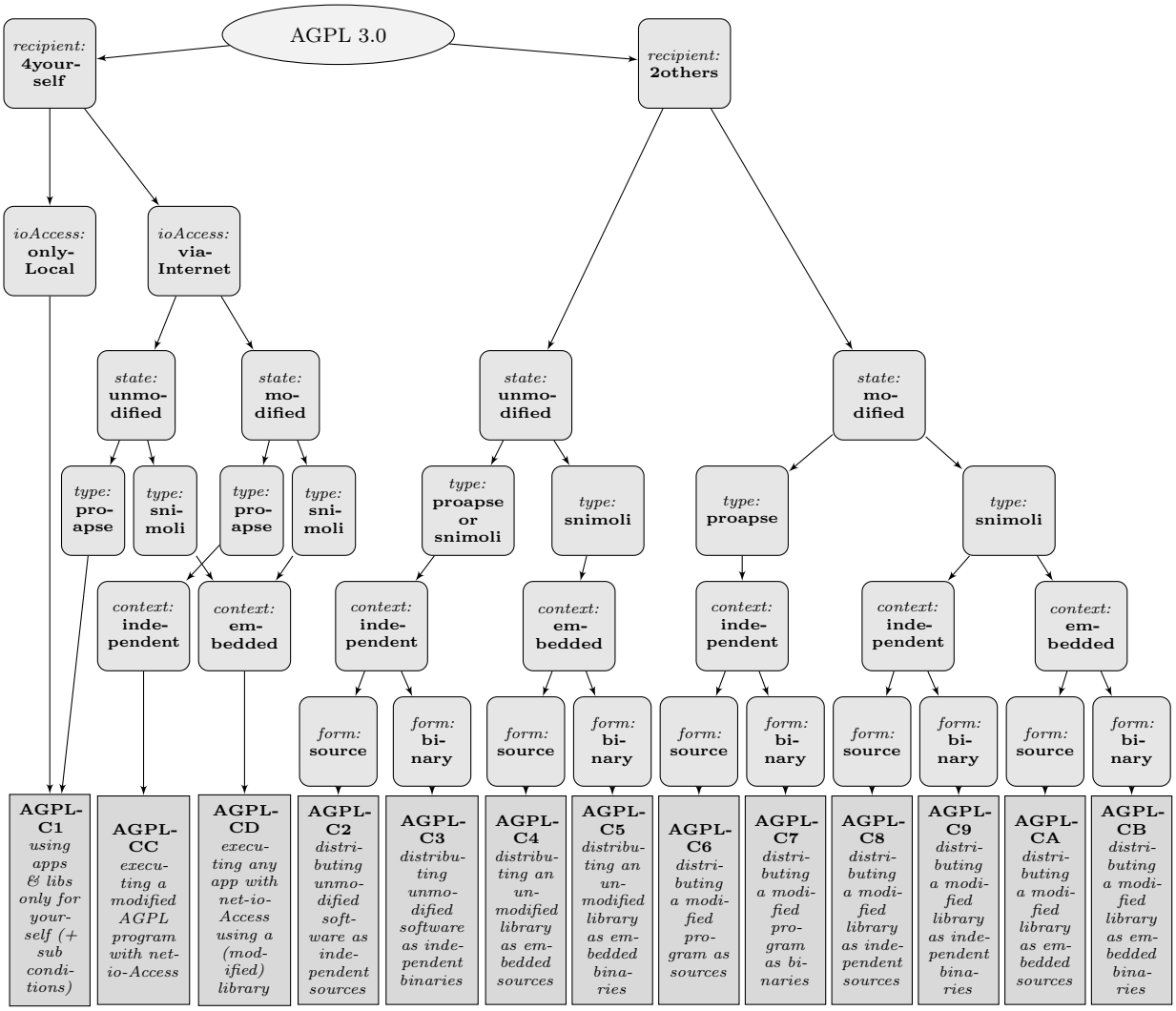
[407] To give someone anything they can't touch, feel or see is like not giving him the object ;-)

[408] Additionally, in the open source community, it is a good tradition, to present these reference data voluntarily.

[409] Moreover, the advantage of doing the job oneself is that one has not to struggle with uncommunicated implicit modifications of the link targets.

that the link will remain valid forever, the point is: you have to fulfill the license, no one else.

6.2 AGPL licensed software



6.2.1 AGPL-3.0-C1: Using the software only for yourself under additional restrictions

means that you received AGPL-3.0 licensed software, that you will use it only for yourself, and that you do not hand over to any third party in any sense. Additionally you warrants that no other than you interacts with the executed software remotely through a computer network.

covers OSUC-01, OSUC-03L, OSUC-06L, and OSUC-09L[410]

requires no tasks in order to fulfill the conditions of the GNU Affero General Public License Version 3 with respect to this use case:

- You are allowed to execute an unmodified AGPL program without being obliged to do anything, as long as you do not give the program to third parties. And you are allowed to embed any AGPL licensed library, snippet or module into your own program and to execute that program without being obliged to do anything, as long as no other than you can interact with it remotely through a computer network and as long as you do not give the library or your program to third parties.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.2 AGPL-3.0-C2: Passing the unmodified software as independent sources

means that you received AGPL-3.0 licensed software that you are now going to distribute to third parties as an independent unit and in the form of unmodified source code files or as an unmodified source code package. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin.

covers OSUC-02S, OSUC-05S[411]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the AGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the AGPL-3.0 license. If it is not already part of the software package, add it.[412]
- [**mandatory:**] Retain all existing copyright notices.

[410] For details → OSLiC, pp. 112 – 123
[411] For details → OSLiC, pp. 112 – 117
[412] For implementing the handover of files correctly → OSLiC, p. 127

- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.3 AGPL-3.0-C3: Passing the unmodified software as independent binaries

means that you received AGPL-3.0 licensed software, which you are now going to distribute to third parties as an independent unit and in the form of unmodified binary files or as an unmodified binary package. In this case it does not matter if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin.

covers OSUC-02B, OSUC-05B[413]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the AGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the AGPL-3.0 license. If it is not already part of the software package, add it.[414]
- [**mandatory:**] Make the source code of the distributed software publicly available (even though you did not modify it): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Retain all existing copyright notices.

[413] For details → OSLiC, pp. 113 – 118
[414] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Execute the to-do list of use case AGPL-3.0-C2 for the source code that you publish.[415]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.4 AGPL-3.0-C4: Passing the unmodified library as embedded sources

means that you received an AGPL-3.0 licensed snippet, module or library that you are now going to distribute to third parties as an embedded component of a larger unit and in the form of unmodified source code files or as an unmodified source code package.

covers OSUC-07S[416]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the AGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the AGPL-3.0 license. If it is not already part of the software package, add it.[417]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the AGPL-3.0 licensed library and that it is itself licensed under the AGPL-3.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.
- [**mandatory:**] Arrange the the sources of the on-top development of the on-top development in a way that they are covered by the AGPL-3.0

[415] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.
[416] For details → OSLiC, pp. 120
[417] For implementing the handover of files correctly → OSLiC, p. 127

licensing statements.

- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.5 AGPL-3.0-C5: Passing the unmodified library as embedded binaries

means that you received an AGPL-3.0 licensed snippet, module or library that you are now going to distribute to third parties as an embedded component of a larger unit and in the form of unmodified binary files or as unmodified binary package.

covers OSUC-07B[418]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the AGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the AGPL-3.0 license. If it is not already part of the software package, add it.[419]
- [**mandatory:**] Make the *complete* source code of the program embedding the library publicly available (and, therefore, also the source code of the library itself): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the AGPL-3.0 licensed library and that it is itself

[418] For details → OSLiC, pp. 121
[419] For implementing the handover of files correctly → OSLiC, p. 127

licensed under the AGPL-3.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

- [**mandatory:**] Arrange the the binaries of the on-top development of the on-top development in a way that they are covered by the AGPL-3.0 licensing statements.
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Execute the to-do list of use case AGPL-3.0-C4 for the source code that you publish.[420]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.6 AGPL-3.0-C6: Passing a modified program as source code

means that you received an AGPL-3.0 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

covers OSUC-04S[421]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the AGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the AGPL-3.0 license. If it is not already part of the software package, add it.[422]

[420] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.
[421] For details → OSLiC, pp. 116
[422] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Let the copyright dialog of the program clearly say that it is a AGPL-3.0 licensed program. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0. If these conditions are not already met, add the missing elements.
- [**mandatory:**] Mark all modifications of the source code the program (proapse) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the program in a way that they are covered by existing AGPL-3.0 licensing statements. If you add new source code files to the program, insert a header containing your copyright line and a licensing statement in the form recommended by the AGPL-3.0.[423]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.7 AGPL-3.0-C7: Passing a modified program as binary

means that you received an AGPL-3.0 licensed program, application, or server (proapse), that you modified it, and that you are now going todistribute this modified version to third parties in the form of binary files or as a binary package.

covers OSUC-04B[424]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the AGPL-3.0 and to the absence of any warranty) are

[423] For details see section 'How to Apply These Terms to Your New Programs' in the AGPL-3.0 license.

[424] For details → OSLiC, pp. 116

retained in your package in the form in which you have received them.

- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the AGPL-3.0 license. If it is not already part of the software package, add it.[425]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Mark all modifications of the source code the program (proapse) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Let the copyright dialog of the program clearly say that it is a AGPL-3.0 licensed program. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0. If these conditions are not already met, add the missing elements.
- [**mandatory:**] Arrange your modifications of the program in a way that they are covered by existing AGPL-3.0 licensing statements. If you add new source code files to the program, insert a header containing your copyright line and a licensing statement in the form recommended by the AGPL-3.0.[426]
- [**mandatory:**] Make the source code of the distributed software publicly available: Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case AGPL-3.0-C6 for the source code that you publish.[427]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright

[425] For implementing the handover of files correctly → OSLiC, p. 127

[426] For details see section 'How to Apply These Terms to Your New Programs' in the AGPL-3.0 license.

[427] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.8 AGPL-3.0-C8: Passing a modified library as independent source code

means that you received an AGPL-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

covers OSUC-08S[428]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the AGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the AGPL-3.0 license. If it is not already part of the software package, add it.[429]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Mark all modifications of the source code of the library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the library in a way that they are covered by existing AGPL-3.0 licensing statements. If you add new source code files to the library, insert a header containing your copyright line and a licensing statement in the form recommended by the AGPL-3.0.[430]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to

[428] For details → OSLiC, pp. 122

[429] For implementing the handover of files correctly → OSLiC, p. 127

[430] For details see section 'How to Apply These Terms to Your New Programs' in the AGPL-3.0 license.

the *modification text file*.

- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.9 AGPL-3.0-C9: Passing a modified library as independent binary

means that you received an AGPL-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers OSUC-08B[431]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the AGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the AGPL-3.0 license. If it is not already part of the software package, add it.[432]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Make the source code of the distributed software publicly available: Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.

[431] For details → OSLiC, pp. 123
[432] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Execute the to-do list of use case AGPL-3.0-C8 for the source code that you publish.[433]
- [**mandatory:**] Mark all modifications of the source code of the library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the library in a way that they are covered by existing AGPL-3.0 licensing statements. If you add new source code files to the library, insert a header containing your copyright line and a licensing statement in the form recommended by the AGPL-3.0.[434]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.10 AGPL-3.0-CA: Passing a modified library as embedded source code

means that you received an AGPL-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10S[435]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the AGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.

[433] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

[434] For details see section 'How to Apply These Terms to Your New Programs' in the AGPL-3.0 license.

[435] For details → OSLiC, pp. 125

- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the AGPL-3.0 license. If it is not already part of the software package, add it.[436]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the AGPL-3.0 licensed library and that it is itself licensed under the AGPL-3.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.
- [**mandatory:**] Mark all modifications of the source code of the embedded library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the embedded library in a way that they are covered by existing AGPL-3.0 licensing statements. If you add new source code files to the embedded library, insert a header containing your copyright line and a licensing statement in the form recommended by the AGPL-3.0.[437]
- [**mandatory:**] Arrange the the sources of the on-top development of the on-top development in a way that they are covered by the AGPL-3.0 licensing statements.
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

[436] For implementing the handover of files correctly → OSLiC, p. 127

[437] For details see section 'How to Apply These Terms to Your New Programs' in the AGPL-3.0 license.

6.2.11 AGPL-3.0-CB: Passing a modified library as embedded binary

means that you received an AGPL-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third partiesin the form of binary files or as a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10B[438]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the AGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the AGPL-3.0 license. If it is not already part of the software package, add it.[439]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Make the *complete* source code of the program embedding the library publicly available (and, therefore, also the source code of the library itself): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case AGPL-3.0-CA for the source code that you publish.[440]
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the AGPL-3.0 licensed library and that it is itself licensed under the AGPL-3.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.
- [**mandatory:**] Mark all modifications of the source code of the embedded library (snimoli) thoroughly within the source code and include

[438] For details → OSLiC, pp. 126

[439] For implementing the handover of files correctly → OSLiC, p. 127

[440] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

the date of the modification.

- [**mandatory:**] Arrange your modifications of the embedded library in a way that they are covered by existing AGPL-3.0 licensing statements. If you add new source code files to the embedded library, insert a header containing your copyright line and a licensing statement in the form recommended by the AGPL-3.0.[441]
- [**mandatory:**] Arrange the the binaries of the on-top development of the on-top development in a way that they are covered by the AGPL-3.0 licensing statements.
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.12 AGPL-3.0-CC: Executing a modified program with network interaction

means that you received an AGPL-3.0 licensed program, an application, or server, that you modified it, and that you let this program, application, or server be executed by a computer in a way, that other people than you can interact with the executed software remotely through a computer network.

covers OSUC-03N[442]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the AGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.

[441] For details see section 'How to Apply These Terms to Your New Programs' in the AGPL-3.0 license.

[442] For details → OSLiC, pp. 115

- [**mandatory:**] Give the recipient a copy of the AGPL-3.0 license. If it is not already part of the software package, add it.[443]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Mark all modifications of the source code the program (proapse) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Let the copyright dialog of the program clearly say that it is a AGPL-3.0 licensed program. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0. If these conditions are not already met, add the missing elements.
- [**mandatory:**] Arrange your modifications of the program in a way that they are covered by existing AGPL-3.0 licensing statements. If you add new source code files to the program, insert a header containing your copyright line and a licensing statement in the form recommended by the AGPL-3.0.[444]
- [**mandatory:**] Make the source code of the executed modified program publicly available: Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case AGPL-3.0-C6 for the source code that you publish.[445]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

prohibits ...

443) For implementing the handover of files correctly → OSLiC, p. 127

444) For details see section 'How to Apply These Terms to Your New Programs' in the AGPL-3.0 license.

445) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.13 AGPL-3.0-CD: Executing a (modified) library as embedded component with network interaction

means that you received an AGPL-3.0 licensed library, snippet, or module, that you modified it or that you did not modified it, that you embed this modified or unm odified library, snippet, or module into an own overarching program, an application, or server, and that you finally let this own program, application, or server be executed by a computer in a way, that other people than you can interact with the executed software remotely through a computer network.

covers OSUC-06N,OSUC-09N[446]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the AGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the AGPL-3.0 license. If it is not already part of the software package, add it.[447]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Make the *complete* source code of the excuted program embedding the (modified) library publicly available (and, therefore, also the source code of the (modified) library itself): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case AGPL-3.0-CA for the source code that you publish.[448]

[446] For details → OSLiC, pp. 120 – 124

[447] For implementing the handover of files correctly → OSLiC, p. 127

[448] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the AGPL-3.0 licensed library and that it is itself licensed under the AGPL-3.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.
- [**mandatory:**] Mark all modifications of the source code of the embedded library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the embedded library in a way that they are covered by existing AGPL-3.0 licensing statements. If you add new source code files to the embedded library, insert a header containing your copyright line and a licensing statement in the form recommended by the AGPL-3.0.[449]
- [**mandatory:**] Arrange the the binaries of the on-top development of the on-top development in a way that they are covered by the AGPL-3.0 licensing statements.
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the AGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.2.14 Discussions and Explanations

For simplifying the justifications of our AGPL interpretation, we can state, that the AGPL-3.0 and the GPL-3.0 are very similar: apart from some differences caused by the varying names and passings remarks[450], the most paragraphs of the two licenses exactly offer the same text[451]. Only the §13 of the AGPL-3.0 does not match to the §13 of the GPL-3.0: §13 of the GPL-3.0 permits "[...] to link or

[449] For details see section 'How to Apply These Terms to Your New Programs' in the AGPL-3.0 license.

[450] Very similar are the preamble and §0. Compare *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. versus *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, wp.

[451] Equal are §1 - 12 and §14 - §17. Compare *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. versus *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, wp.

combine any covered work with a work lincesed under version 3 of the GNU Affero Generasl Public License"[452]; while §13 of the AGPL-3.0 deals with the "remote network interaction"[453]. Therefore, the analysis of the GPL-3.0 lincense[454] is also valid for the AGPL-3.0; it is not necessary to repeat that discussion here.

So, we can focus on the difference. The AGPL-3.0 tries to close a gap of the GPL-3.0:

Purpose of all GNU licenses is to preserve the freedom to use, to study, to share, and to modify the GNU programs and libraries[455]. These licenses want to prevent that users circumvent the tasks which establish and maintain this freedom: Only if someone uses the program / library only for himself, he shall not be obliged to do anything. But if any third party was involved into the use of the GNU software, this third party should receive all those rights and possibilities to use the software which all the other users already have got.

In a time, where using the benefits of a program meant *executing the software on ons's own machine* (and hence *having received the program at least as a binary*), it was enough to let the obligations of – for example – handing over the license or the source code be triggered by the act of 'distributing the software'. Nowadays, in the times of cloud software systems, users can let profit other users from the free software without conveying the software. In these cases, they execute the free program on their own machines, but they nevertheless do not use the free program any longer only for themselves. So, in time of cloud service technologies, the trigger of executing the license fulfilling tasks must be complemented by a criterion which indicates that a third party is involved into the context of using the software. And this criteroin must no longer presuppose that this third party has received the software itself.

For that purpose, the AGPL-3.0 states, that such an executed AGPL program must "[...] prominently offer all useres *interacting with it remotely through a computer network* [...] an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software"[456]. Obviously, the trigger of *distributing the AGPL software* now has been expanded by the feature *being able to interact with the AGPL software remotely through a computer network*.

The first consequence of this analysis is, that we can take over all the GPL uses

[452] cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. §13.
[453] cf. *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, wp. §13.
[454] → p. 236
[455] cf. *Free Software Foundation*: What is free software? The Free Software Definition; 2015 [n.y.] ⟨URL: https://www.gnu.org/philosophy/free-sw.en.html⟩ – reference download: 2015-02-20, wp..
[456] cf. *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, wp. §13.

cases which deal with *distributing the software* (2others) and all the corresponding license fulfilling tasklists of GPL-3-C2[457] until GPL-3-CB[458] – as we have defined them in the GPL chapter.

The second consequence is, that we now have to subclassify the open source use case *recipient:4yourself*: we have to distinguish the use with internet input-output access from that with only local input-output acesss.

Additionally, the AGPL limits the requirement to the condition, that the used program is modified. The license exactly says that "[...] if you *modify* the Program, your modfied version must prominently offer all useres *interacting with it remotely through a computer network* [...] an opportunity to receive the Corresponding Source of your version [...]"[459]. Thus, the third consequence is, that we have to subclassify the open source use case *recipient:4yourself* not only by the features *ioAccess:viaInternet* and *ioAccess:onlyLocal*, but also by the features *state:modified* and *state:unmodified*.

Finally, there is another little complication: One can only execute a program. A library can not be directly executed. So, the question arises, what the user has to be do if executes an own program which uses an unmodified AGPL licensed library or module?

On the first glance, the license in §13 says only that he has to publish the sources too, if executes a modified program. But on further reflection, one has also to consider the other paragraphs of the AGPL: If one embeds an AGPL licensed library, snippet or module into an own program, then – due to the Copyleft effect of the AGPL – this program which uses the library, snippet or module, has to be licensed under the AGPL too. And finally, every new program has to be regarded as a modification of the first empty file. In other words: one can only execute an own program using an unmodified AGPL library compliantly, if one respects the §13 for the complete software complex being comprised of the library itself and the pure code of the overarching program.

Based on this analysis, we had only to introduce two new AGPL specific open source use cases and could recycle the complete set of GPL specific open source use cases:

- All GPL-3.0 use cases triggered by the distribution of the software *recipient:2others* are transfered into the AGPL-3.0 finder and the AGPL-3.0 tasklist chapter as they have been defined in the GPL finder and the GPL-3.0 tasklist chapter.
- All combinations of *recipient:4yourself* and *ioAccess:onlyLocally* are covered by the old GPL 'yourself' use case which says, that one has not do anything

[457] → OSLiC, p. 224
[458] → OSLiC, p. 235
[459] cf. *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, wp. §13.

as long as one uses the software only for oneself. But in the context of AGPL, this use case has additional conditions: one has not do anything if one does not distribute the software to other parties in any thing and if one executes this software on one's own machines in an environment which does not allow anyone else than oneself to interact with it remotely through a computer network.

- If one executes an unmodified AGPL program, which one has received and which one has not modified, then one also has not do anything.
- If one 'executes' an unmodified library as an embedded component of the really executed overarching program, then one has also to license this overarching program under the AGPL and hence has to fulfill the conditions of §13.
- If one executes a modified AGPL program, which one has received, has to fulfill the conditions of §13.
- If one executes an modified library as an embedded component of the really executed overarching program, then one has also to license this overarching program under the AGPL and hence has to fulfill the conditions of §13 with respect to bot parts, to the overarching program and the library.

There is a last point, which should also be discussed here. It concerns the question of granularity:

The AGPL-3.0 requires that the "[...] modified version (of an [executed] program) must prominently offer all users interacting with it remotely through a computer network [...] an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge [...]"[460]. For respecting this rule, one has to know what the term *Corresponding Source* means: how many of the embedded components of the program must be conveyed together with the overarching program.

Fortunately, the AGPL-3.0 (and the GPL-3.0) defines the used terms: "The 'Corresponding Source' for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities.[461]" If one took this statements seriously, one would have to "provide access to" the complete software stack of the executed AGPL program – just down to the glibc.

But the AGPL does not want to be to greedy. Therefore it limits the scope by determining, that the *Corresponding Source* "[...] does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part

460) cf. *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, wp. §13.
461) cf. id., l.c., wp. §1.

of the work"[462]. For understanding this rule, one has to know, what the term *System Libraries* means. The AGPl says, that "the 'System Libraries' of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form.[463]" Unfortunately, one has now to analyse, what the AGPL defines as a *Major Component*: "A enquoteMajor Component, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it[464]".

Based on these specifications, one can give some rule of thumbs concerning the question down to which level one has to give access to the corresponding source code of an an executed AGPL program:

- If one lets execute a modified AGPL licensed binary program, then one has to give access to the code of
  - the executed program itself
  - every modified embedded component of that program
  - every not freely accessible embedded component of that program
  - all not freely accessible tools, scripts, data which are necessary to compile the sources of the program in a freely accessible compilation / developement environment

  But it is not necessary to give access to unmodified standard libraries, compilers, or tools which can freely be downloaded from their standard repositories.
- If one lets execute a modified AGPL licensed script, then one has to give access to the code of
  - the executed script itself
  - every modified embedded script component included by the main script
  - every not freely accessible embedded script component included by the main script
  - all not freely accessible tools, scripts, data which are necessary to to let that main script be executed by a freely accessible interpreter

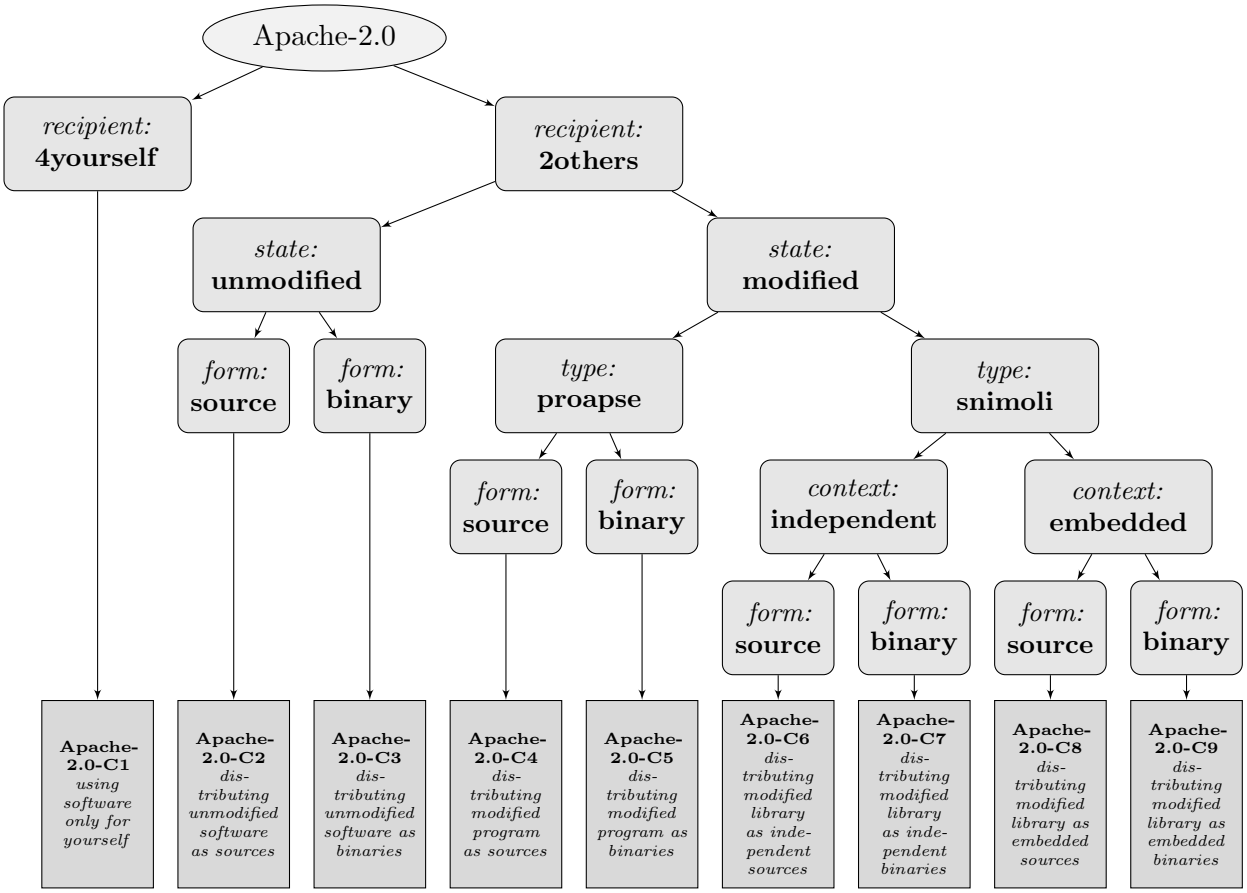
[462] cf. *Open Source Initiative*: The AGPL-3.0 License (OSI), 2007, wp. §1.
[463] cf. id., ibid.
[464] cf. id., ibid.

– the interpreter itself if it is not freely accessible.

But it is not necessary to give access to unmodified standard script libraries, interpreters, or tools which can freely be downloaded from their standard repositories

6.3 Apache-2.0 licensed software

Today, the current release of the Apache open source license is version 2.0, older versions are deprecated.[465] Because it focusses primarily on the "redistribution,"[466] the following simplified Apache specific open source use case finder[467] can be used:



[465] For details → OSLiC, pp. 29
[466] cf. *Open Source Initiative*: APL-2.0, 2004, wp. §4.
[467] For details of the general OSUC finder → OSLiC, pp. 104 and **??**

6.3.1 Apache-2.0-C1: Using the software only for yourself

means that you received Apache-2.0 licensed software, that you will use it only for yourself, and that you do not hand it over to any 3rd party in any sense.

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L, and OSUC-09N[468]

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

- You are allowed to use any kind of Apache software in any sense and in any context without being obliged to do anything as long as you do not give the software to third parties.

prohibits ...

- to promote any of your services based on the this software by trademarks, service marks, or product names linked to the software except as required for reasonable and customary use in describing the software file.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.3.2 Apache-2.0-C2: Passing the unmodified software as source code

means that you received Apache-2.0 licensed software which you are now going to distribute to third parties in the form of unmodified source code files or as unmodified source code package. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or as an embedded unit.

covers OSUC-02S, OSUC-05S, OSUC-07S[469]

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 not already part of the software package, add it.[470]
- **[mandatory:]** Ensure that the licensing elements (especially the specific copyright notice of the original author(s)) are retained in your package in the form you have received them.

[468] For details → OSLiC, pp. 112 – 124
[469] For details → OSLiC, pp. 112 – 120
[470] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Ensure that the *notice text file* is retained in your package in the form you have initially received it.[471]
- [**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 promote any of your services based on the this software by trademarks, service marks, or product names linked to the software except as required for reasonable and customary use in describing the software file.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.3.3 Apache-2.0-C3: Passing the unmodified software as binaries

means that you received Apache-2.0 licensed software which you are now going to distribute to third parties in the form of unmodified binary files or as unmodified binary package. In this case it does not matter if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit.

covers OSUC-02B, OSUC-05B, OSUC-07B[472]

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 not already part of the binary package, add it.[473]
- [**mandatory:**] Ensure that the licensing elements (especially 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 in or integrated into your package in the form you have initially received

[471] The Apache license seems purposely to be a bit ambiguous: 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. To be sure to act according to this requirement you should also read this term in the broader sense if there is no text file named 'NOTICE'

[472] For details → OSLiC, pp. 113 – 121

[473] For implementing the handover of files correctly → OSLiC, p. 127

it.

- [**mandatory:**] Ensure that the *notice text file* is also reproduced if and whereever such third-party notices normally appear (especially, if you are distributing an unmodified Apache-2.0 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 a subsection of your own copyright notice.

prohibits ...

- to promote any of your services based on the this software by trademarks, service marks, or product names linked to the software except as required for reasonable and customary use in describing the software file.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.3.4 Apache-2.0-C4: Passing a modified program as source code

means that you received an Apache-2.0 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

covers OSUC-04S[474]

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 not already part of the software package, add it.[475]
- [**mandatory:**] Ensure that the licensing elements (especially 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 in the *notice text file* that you have received.
- [**mandatory:**] Ensure that the *notice text file* is also reproduced if and whereever such third-party notices normally appear. If the program already displays a copyright dialog, update it in an appropriate manner.

[474] For details → OSLiC, pp. 116
[475] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Inside of the source code, mark all your modifications thoroughly. Generate a *notice text file*, if it still does not exist. *Add* a description of your modifications into the *notice text file.*
- [**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 promote any of your services based on the this software by trademarks, service marks, or product names linked to the software except as required for reasonable and customary use in describing the software file.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.3.5 Apache-2.0-C5: Passing a modified program as binary

means that you received an Apache-2.0 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package.

covers OSUC-04B[476]

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 not already part of the binary package, add it.[477]
- [**mandatory:**] Ensure that the licensing elements (especially 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 in the *notice text file* that you have received. Create a *notice text file*, if it still does not exist. *Add* a description of your modifications into the *notice text file.*

[476] For details → OSLiC, pp. 116
[477] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Ensure that the *notice text file* is also reproduced if and whereever such third-party notices normally appear. If the program already displays a copyright dialog, update it in an appropriate manner.
- [**voluntary:**] Even if you do not want to distribute your modified source code, mark all your modifications thoroughly.
- [**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 promote any of your services based on the this software by trademarks, service marks, or product names linked to the software except as required for reasonable and customary use in describing the software file.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.3.6 Apache-2.0-C6: Passing a modified library as independent source code

means that you received an Apache-2.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

covers OSUC-08S[478]

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 not already part of the software package, add it.[479]
- [**mandatory:**] Ensure that the licensing elements (especially 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 in the *notice text file* that you have received.

[478] For details → OSLiC, pp. 122
[479] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Inside of the source code, mark all your modifications thoroughly. Generate a *notice text file*, if it still does not exist. *Expand* 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 promote any of your services based on the this software by trademarks, service marks, or product names linked to the software except as required for reasonable and customary use in describing the software file.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.3.7 Apache-2.0-C7: Passing a modified library as independent binary

means that you received an Apache-2.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers OSUC-08B[480]

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 not already part of the binary package, add it.[481]
- [**mandatory:**] Ensure that the licensing elements (especially 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 in the *notice text file* that you have received. Create a *notice text file*, if it still does not exist. *Add* a description of your modifications into the *notice text file.*

[480] For details → OSLiC, pp. 123
[481] For implementing the handover of files correctly → OSLiC, p. 127

- [**voluntary:**] Even if you do not want to distribute your modified source code, mark all your modifications thoroughly.
- [**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 promote any of your services based on the this software by trademarks, service marks, or product names linked to the software except as required for reasonable and customary use in describing the software file.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.3.8 Apache-2.0-C8: Passing a modified library as embedded source code

means that you received an Apache-2.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10S[482]

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 not already part of the software package, add it.[483]
- [**mandatory:**] Ensure that the licensing elements (especially 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 in the *notice text file* that you have received.
- [**mandatory:**] Ensure that the *notice text file* is also reproduced if and whereever such third-party notices normally appear. If the software that embeds this library displays its own copyright dialog, insert this information there.

[482] For details → OSLiC, pp. 125
[483] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Inside of the library source code, mark all your modifications thoroughly. Generate a *notice text file*, if it still does not exist. *Expand* the *notice text file* by a description of your modifications.[484]
- [**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 embedded components like libraries, modules, snippets, or plugins in a specific directory which contains also all additional licensing elements.

prohibits ...

- to promote any of your services based on the this software by trademarks, service marks, or product names linked to the software except as required for reasonable and customary use in describing the software file.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.3.9 Apache-2.0-C9: Passing a modified library as embedded binary

means that you received an Apache-2.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10B[485]

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 not already part of the binary package, add it.[486]
- [**mandatory:**] Ensure that the licensing elements (especially 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

[484] The term library also includes snippet, module, and plugin.
[485] For details → OSLiC, pp. 126
[486] For implementing the handover of files correctly → OSLiC, p. 127

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 in the *notice text file* that you have received. Create a *notice text file*, if it still does not exist. *Add* a description of your modifications into the *notice text file.*
- [**mandatory:**] Ensure that the *notice text file* is also reproduced if and whereever such third-party notices normally appear. If the software that embeds this library displays its own copyright dialog, insert this information there.
- [**voluntary:**] Even if you do not want to distribute your modified source code, mark all your modifications of the embedded libary thoroughly. [487]
- [**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.
- [**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 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 promote any of your services based on the this software by trademarks, service marks, or product names linked to the software except as required for reasonable and customary use in describing the software file.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.3.10 Discussions and Explanations

- On the one hand, the Apache 2.0 license does not permit "[...] to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file"[488]. On the

[487] library or snippet, or module, or plugin
[488] cf. *Open Source Initiative*: APL-2.0, 2004, wp. §6.

other hand, this license alerts that all the patent licenses granted to those who "[...] institute a patent litigation" will terminate automatically[489]. Hence, the OSLiC generally (Apache-2.0-C1 - Apache-2.0-C9) interdicts to promote products or services by these elements and to legally fight against patents linked to the software.

- The Apache-2.0 also requires to "[...] give any other recipients of the Work or Derivative Works a copy of this License"[490]. Therefore, all *2others* use cases contain the respective mandatory condition (Apache-2.0-C2 - Apache-2.0-C9).
- Additionally, the Apache-2.0 requires, that modifications must be marked[491]. Thus, in all cases of passing the modified software in the form of source code the OSLiC requires to mark the modifications and to integrate a hint into the notice file—while in all the cases of passing the modified software in the form of binaries it inserts only a voluntary condition (Apache-2.0-C4 - Apache-2.0-C9).
- Furthermore, the Apache-2.0 requires that one must "[...] retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work" So, the OSLIC requires in all contexts (Apache-2.0-C1 - Apache-2.0-C9) that the licensing elements are retained in the form you have received them[492].
- Finally, the Apache-2.0 requires that the received "NOTICE text file" must be integrated as readable copy to each package distributed in the form of source code, or—in case of binary distibutions—must be displayed "[...] if and wherever such third-party notices normally appear"[493]. Thus, the OSLiC requires mandatorily that all source code distributions must include the notice text file (Apache-2.0-C2, Apache-2.0-C4, Apache-2.0-C6, Apache-2.0-C8) and that all distributions of binary applications which normally show such a copyrigth screen must integrate the content of the notice file into this screen (Apache-2.0-C5, Apache-2.0-C9). For libraries distributed in the form of binaries it is assumed that they normally do not contain such copyright dialogs (Apache-2.0-C7)

489) cf. *Open Source Initiative*: APL-2.0, 2004, wp. §3.
490) cf. id., l.c., wp. §4.1.
491) cf. id., l.c., wp. §4.2.
492) This might confuse some readers: Yes, even if you distribute a modified version in the form of binaries you must fulfill this condition. Moreover, you must also hand the license over to your receipient. But, nevertheless, you are not obliged to publish the modified source code, too. (→ OSLiC, p. 29)
493) cf. id., l.c., wp. §4.4.

6.4 BSD licensed software

As an approved open source license, the BSD license exists in two versions[494] The latest release is the *BSD 2-Clause license,*[495], the older release is the *BSD 3-Clause license.*[496] The very little differences between the two versions have to be respected exactly.

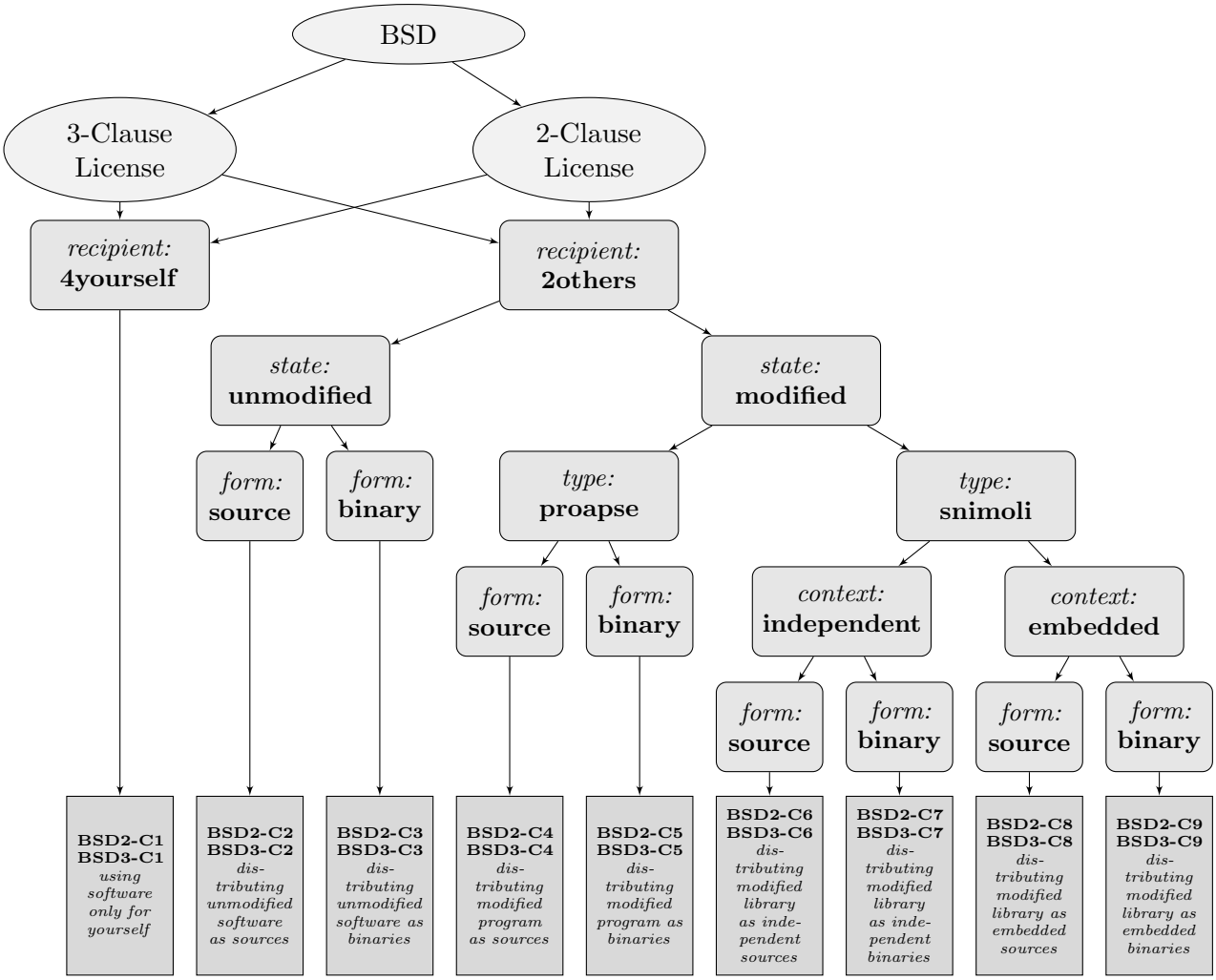
All BSD open source licenses focus explicitely on the (re-)distribution *open source use cases,* which we have specified by our token *2others*. Conditions for the other use cases specified by the token *4yourself* can be derived.[497] Additionally the BSD licenses distinguishes between different forms of distribution, esp. whether the work is distributed as a (set of) source code file(s) or as a set of binary file(s). Use the following tree to find the BSD license fulfilling to-do lists.

494) 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". Because of that cancellation you can simply act according the cf. *Open Source Initiative*: The BSD 3-Clause License, 2012, wp. if you have to fulfill the oldest of the BSD licenses.

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

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

497) For details of the *open source use case tokens* see p. 104. For details of the *open source use cases* based on these token see p. **??**



6.4.1 BSD-3-Clause-C1: Using the software only for yourself

means that you received BSD licensed software, that you will use it only for yourself and that you do not hand it over to any 3rd party in any sense.

covers

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L and OSUC-09N[498]

requires no tasks in order to fulfill the conditions of the New BSD (3 Clauses) 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 as long as you do not give the software to 3rd parties.

[498] For details → OSLiC, pp. 112 – 124

prohibits ...

- to use the name of the licensing organization or the names of the licensing contributors to promote your own work.[499].

6.4.2 BSD-3-Clause-C2: Passing the unmodified software as source code

means that you received BSD licensed software which you are now going to distribute to third parties in the form of unmodified source code files or as unmodified source code package. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or as an embedded unit.

covers

covers OSUC-02S, OSUC-05S, OSUC-07S[500]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly 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 or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.

prohibits ...

- to use the name of the licensing organization or the names of the licensing contributors to promote your own work.

6.4.3 BSD-3-Clause-C3: Passing the unmodified software as binary

means that you received BSD licensed software which you are now going to distribute to third parties in the form of unmodified binary files or as unmodified binary package. In this case it does not matter if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit.

covers

covers OSUC-02B, OSUC-05B, OSUC-07B[501]

[499] which may be, for example, an internet service based on this BSD software used in your own data center

[500] For details → OSLiC, pp. 112 – 120

[501] For details → OSLiC, pp. 113 – 121

requires the following 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 build the binary package from the source code package and if this does not automatically generate and integrate the licensing files then create the copyright notice, the BSD conditions, and the BSD disclaimer in the form found to the in the source code package and insert these files into your distribution manually.[502]
- [**mandatory:**] Let the documentation of your distribution or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.

prohibits ...

- to use the name of the licensing organization or the names of the licensing contributors to promote your own work.

6.4.4 BSD-3-Clause-C4: Passing a modified program as source code

means that you received a BSD licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

covers

covers OSUC-04S[503]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly 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 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 message that 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.

[502] For implementing the handover of files correctly → OSLiC, p. 127
[503] For details → OSLiC, pp. 116

prohibits ...

- to use the name of the licensing organization or the names of the licensing contributors to promote your own work.

6.4.5 BSD-3-Clause-C5: Passing a modified program as binary

means that you received a BSD licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package.

covers

covers OSUC-04B[504]

requires the following 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 build the binary package from the source code package and if this does not automatically generate and integrate the licensing files then create the copyright notice, the BSD conditions, and the BSD disclaimer in the form found to the in the source code package and insert these files into your distribution manually.[505]
- [**mandatory:**] Let the documentation of your distribution 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 message that 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 ...

- to use the name of the licensing organization or the names of the licensing contributors to promote your own work.

6.4.6 BSD-3-Clause-C6: Passing a modified library as independent source code

means that you received a BSD licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute

[504] For details → OSLiC, pp. 116

[505] For implementing the handover of files correctly → OSLiC, p. 127

this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

covers

covers OSUC-08S[506]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly 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 or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.

prohibits ...

- to use the name of the licensing organization or the names of the licensing contributors to promote your own work.

6.4.7 BSD-3-Clause-C7: Passing a modified library as independent binary

means that you received a BSD licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers

covers OSUC-08B[507]

requires the following 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 build the binary package from the source code package and if this does not automatically generate and integrate the licensing files then create the copyright notice, the BSD conditions, and the BSD disclaimer in the form found to the in the source code package and insert these files into your distribution manually.[508]

[506] For details → OSLiC, pp. 122
[507] For details → OSLiC, pp. 123
[508] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Let the documentation of your distribution or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.

prohibits ...

- to use the name of the licensing organization or the names of the licensing contributors to promote your own work.

6.4.8 BSD-3-Clause-C8: Passing a modified library as embedded source code

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

covers

covers OSUC-10S[509]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly 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 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 message that 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 licensing elements (particularly, the BSD license text, the copyright notice of the original author(s), and the BSD disclaimer) clearly refer only to the embedded library and do not affect the licensing of your own overarching work. It's a good tradition to keep the embedded

[509] For details → OSLiC, pp. 125

components like libraries, modules, snippets, or plugins in separate directories, which also contains all their licensing elements.

prohibits ...

- to use the name of the licensing organization or the names of the licensing contributors to promote your own work.

6.4.9 BSD-3-Clause-C9: Passing a modified library as embedded binary

means that you received a BSD licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers

covers OSUC-10B[510]

requires the following 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 build the binary package from the source code package and if this does not automatically generate and integrate the licensing files then create the copyright notice, the BSD conditions, and the BSD disclaimer in the form found to the in the source code package and insert these files into your distribution manually.[511]
- [**mandatory:**] Let the documentation of your distribution 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 message that 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 (particularly, the BSD license text, the copyright notice of the original author(s), and the BSD disclaimer) clearly refer only to the embedded library and do not affect the licensing of your own overarching

[510] For details → OSLiC, pp. 126
[511] For implementing the handover of files correctly → OSLiC, p. 127

work. It's a good tradition to keep the embedded components like libraries, modules, snippets, or plugins in separate directories, which also contains all their licensing elements.

prohibits ...

- to use the name of the licensing organization or the names of the licensing contributors to promote your own work.

6.4.10 BSD-2-Clause-C1: Using the software only for yourself

means that you received BSD licensed software, that you will use it only for yourself and that you do not hand it over to any 3rd party in any sense.

covers

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L and OSUC-09N[512]

requires no tasks in order to fulfill the conditions of the Simplified BSD (2 Clauses) 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 as long as you do not give the software to 3rd parties.

prohibits nothing explicitely.

6.4.11 BSD-2-Clause-C2: Passing the unmodified software as source code

means that you received BSD licensed software which you are now going to distribute to third parties in the form of unmodified source code files or as unmodified source code package. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or as an embedded unit.

covers

covers OSUC-02S, OSUC-05S, OSUC-07S[513]

requires the following tasks in order to fulfill the license conditions:

- **[mandatory:]** Ensure that the licensing elements (particularly 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.

[512] For details → OSLiC, pp. 112 – 124
[513] For details → OSLiC, pp. 112 – 120

- [**voluntary:**] Let the documentation of your distribution or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.

prohibits nothing explicitely.

6.4.12 BSD-2-Clause-C3: Passing the unmodified software as binary

means that you received BSD licensed software which you are now going to distribute to third parties in the form of unmodified binary files or as unmodified binary package. In this case it does not matter if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit.

covers

covers OSUC-02B, OSUC-05B, OSUC-07B[514]

requires the following 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 build the binary package from the source code package and if this does not automatically generate and integrate the licensing files then create the copyright notice, the BSD conditions, and the BSD disclaimer in the form found to the in the source code package and insert these files into your distribution manually.[515]
- [**mandatory:**] Let the documentation of your distribution or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.

prohibits nothing explicitely.

6.4.13 BSD-2-Clause-C4: Passing a modified program as source code

means that you received a BSD licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

covers

covers OSUC-04S[516]

[514] For details → OSLiC, pp. 113 – 121
[515] For implementing the handover of files correctly → OSLiC, p. 127
[516] For details → OSLiC, pp. 116

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly 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 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 message that 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 explicitely.

6.4.14 BSD-2-Clause-C5: Passing a modified program as binary

means that you received a BSD licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package.

covers

covers OSUC-04B[517]

requires the following 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 build the binary package from the source code package and if this does not automatically generate and integrate the licensing files then create the copyright notice, the BSD conditions, and the BSD disclaimer in the form found to the in the source code package and insert these files into your distribution manually.[518]
- [**mandatory:**] Let the documentation of your distribution 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 message that is shown by the running program also state that the program is licensed under the BSD license. Because you

[517] For details → OSLiC, pp. 116
[518] For implementing the handover of files correctly → OSLiC, p. 127

are already modifying the program you can also add such a hint if the presented original copyright notice lacks such a statement.

prohibits nothing explicitely.

6.4.15 BSD-2-Clause-C6: Passing a modified library as independent source code

means that you received a BSD licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

covers

covers OSUC-08S[519]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly 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 or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.

prohibits nothing explicitely.

6.4.16 BSD-2-Clause-C7: Passing a modified library as independent binary

means that you received a BSD licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers

covers OSUC-08B[520]

requires the following 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

[519] For details → OSLiC, pp. 122
[520] For details → OSLiC, pp. 123

you have received them. If you build the binary package from the source code package and if this does not automatically generate and integrate the licensing files then create the copyright notice, the BSD conditions, and the BSD disclaimer in the form found to the in the source code package and insert these files into your distribution manually.[521]

- [**mandatory:**] Let the documentation of your distribution or your additional material also contain the original copyright notice, the BSD conditions, and the BSD disclaimer.

prohibits nothing explicitely.

6.4.17 BSD-2-Clause-C8: Passing a modified library as embedded source code

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

covers

covers OSUC-10S[522]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly 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 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 message that 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 licensing elements (particularly, the BSD license text, the copyright

[521] For implementing the handover of files correctly → OSLiC, p. 127
[522] For details → OSLiC, pp. 125

notice of the original author(s), and the BSD disclaimer) clearly refer only to the embedded library and do not affect the licensing of your own overarching work. It's a good tradition to keep the embedded components like libraries, modules, snippets, or plugins in separate directories, which also contains all their licensing elements.

prohibits nothing explicitely.

6.4.18 BSD-2-Clause-C9: Passing a modified library as embedded binary

means that you received a BSD licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers

covers OSUC-10B[523]

requires the following 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 build the binary package from the source code package and if this does not automatically generate and integrate the licensing files then create the copyright notice, the BSD conditions, and the BSD disclaimer in the form found to the in the source code package and insert these files into your distribution manually.[524]
- [**mandatory:**] Let the documentation of your distribution 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 message that 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 (particularly, the BSD license text, the copyright notice of the original author(s), and the BSD disclaimer) clearly refer only to the

[523] For details → OSLiC, pp. 126
[524] For implementing the handover of files correctly → OSLiC, p. 127

embedded library and do not affect the licensing of your own overarching work. It's a good tradition to keep the embedded components like libraries, modules, snippets, or plugins in separate directories, which also contains all their licensing elements.

prohibits nothing explicitely.

6.4.19 Discussions and Explanations

The *BSD 2-Clause license* has a simple structure: In the beginning, it generally "(permits) redistribution and use in source and binary forms, with or without modification, [...]", if one fulfills the two rules of the license.[525] The first rule concerns the (re)distribution in the form of source code, the second the (re)distribution of binary packages. Here are some explanations why we translated the rules into different 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."[526] Hence, you are not allowed to modify any of the copyright notes which are already embedded in the (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*[527]. 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 (particularly 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 the 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."[528] Hence, this is

[525] cf. *Open Source Initiative*: The BSD 2-Clause License, 2012, wp.

[526] cf. id., ibid.

[527] 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 missing 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 BSD licensed software.

[528] cf. id., l.c., wp.

not required as a necessary condition for the (re)distribution as source code package. But nevertheless, even for a distribution in the form of source code, it is often possible to fulfill this rule, too—e.g., if you offer your own download site for source code packages. In such cases, it is a sign of respect to mention the licensing not only inside the packages, but also in the text of your site. Because of that, we added the following voluntary task for all BSD open source use cases which deal with the redistribution in the form of source code:

> *Let the documentation of your distribution 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",[529] we had to rewrite the facultative task for a distribution in the form of source code as a mandatory task for all BSD open source use cases which deals with the redistribution in binary form.
- In case of (re)distributing the program in the 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 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 build the binary package from the source code package and if this does not automatically generate and integrate the licensing files then create the copyright notice, the BSD conditions, and the BSD disclaimer in the form found to the in 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 open source software used and their licenses as part 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

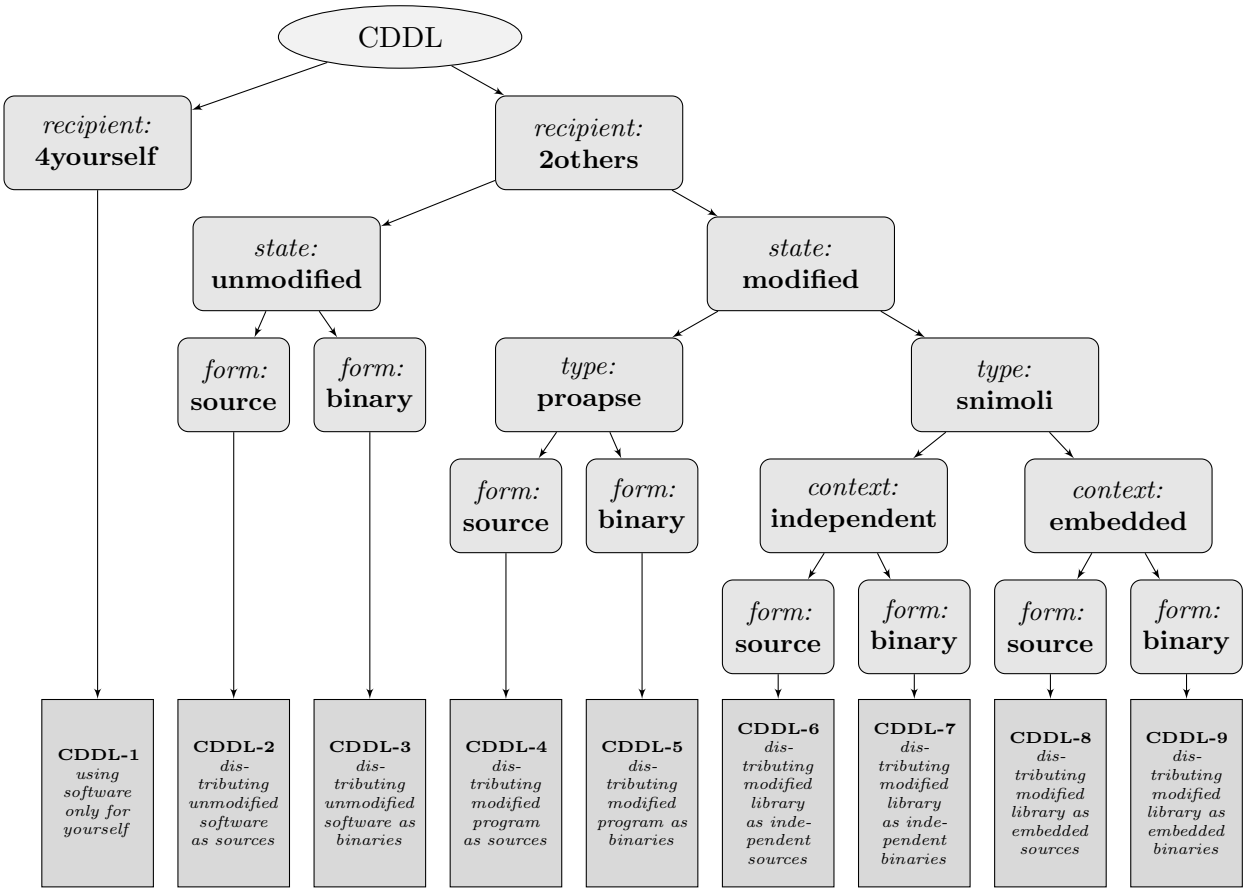
[529] cf. *Open Source Initiative*: The BSD 2-Clause License, 2012, wp.

it, you are also allowed to insert such references, even if they are missing. Therefore we added a third voluntary task to honor this tradition for all relevant open source use cases.

6.5 CDDL licensed software [tbd]

Also, [...]

Thus, for finding the relevant, simply processable task lists, also the following CDDL specific open source use case structure[530] can be used:



6.5.1 CDDL-1: Using the software only for yourself

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

covers OSUC-01, OSUC-03, OSUC-06, and OSUC-09[531]

[530] For details of the general OSUC finder → OSLiC, pp. 104 and **??**

[531] For details → OSLiC, pp. 112 - **??**

requires ...

prohibits ...

6.5.2 CDDL-2: Passing the unmodified software as source code

means that you are going to distribute an unmodified version of the received CDDL software to 3rd parties - in the form of source code files or as a source code package. In this case it is not discriminating to distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit

covers OSUC-02S, OSUC-05S, OSUC-07S[532]

requires the following tasks in order to fulfill the license conditions:

- ...

prohibits ...

- ...

6.5.3 CDDL-3: Passing the unmodified software as binaries

means that you are going to distribute an unmodified version of the received CDDL software to 3rd parties – in the form of binary files or as a binary package. In this case it is not discriminating to distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit.

covers OSUC-02B, OSUC-05B, OSUC-07B[533]

requires the following tasks in order to fulfill the license conditions:

- ...

prohibits ...

- ...

6.5.4 CDDL-4: Passing a modified program as source code

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

[532] For details → OSLiC, pp. 112 - 120
[533] For details → OSLiC, pp. 113 - 121

covers OSUC-04S[534]

requires the following tasks in order to fulfill the license conditions:

- ...

prohibits ...

- ...

6.5.5 CDDL-5: Passing a modified program as binary

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

covers OSUC-04B[535]

requires the following tasks in order to fulfill the license conditions:

- ...

prohibits ...

- ...

6.5.6 CDDL-6: Passing a modified library as independent source code

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

covers OSUC-08S[536]

requires the following tasks in order to fulfill the license conditions:

- ...

prohibits ...

- ...

6.5.7 CDDL-7: Passing a modified library as independent binary

means that you are going to distribute a modified version of the received CDDL licensed code snippet, module, library, or plugin (snimoli) to 3rd parties –

[534] For details → OSLiC, pp. 116
[535] For details → OSLiC, pp. 116
[536] For details → OSLiC, pp. 122

in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers OSUC-08B[537]

requires the following tasks in order to fulfill the license conditions:

- ...

prohibits ...

- ...

6.5.8 CDDL-8: Passing a modified library as embedded source code

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

covers OSUC-10S[538]

requires the following tasks in order to fulfill the license conditions:

- ...

prohibits ...

- ...

6.5.9 CDDL-9: Passing a modified library as embedded binary

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

covers OSUC-10B[539]

requires the following tasks in order to fulfill the license conditions:

- ...

prohibits ...

- ...

[537] For details → OSLiC, pp. 123
[538] For details → OSLiC, pp. 125
[539] For details → OSLiC, pp. 126

6.5.10 Discussions and Explanations

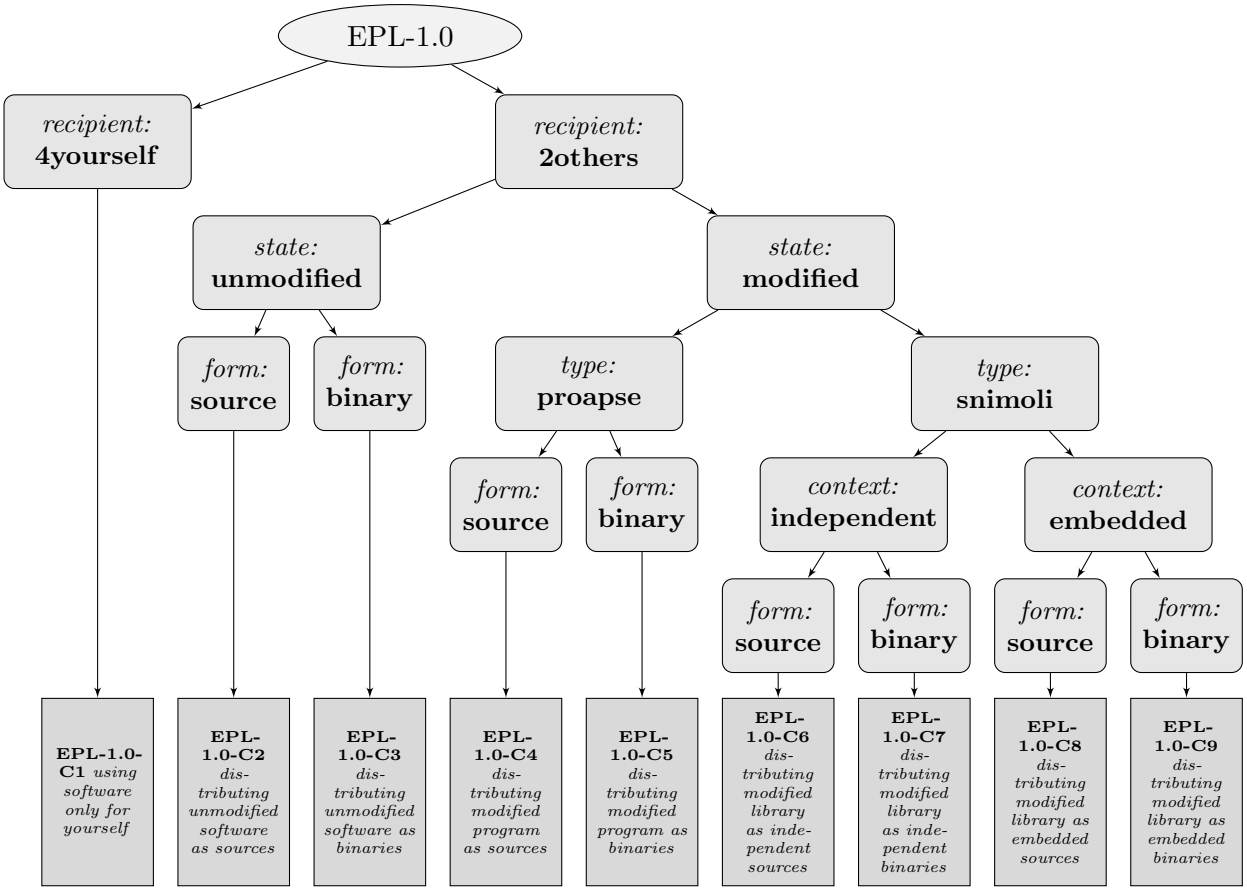
The CDDL offers . . . which contains nearly all requirements[540]. Only for some

-

6.6 EPL-1.0 licensed software

The Eclipse Public License clearly distinguishes the distribution in the form of source code from that in the form of binaries: First, it allows to "distribute" Eclipse licensed programs "in source code and in object code".[541] Then it specifies under which conditions one may distribute the program as a set of binaries.[542] One of these conditions is—roughly speaking—that the distributor makes the sources available too.[543] More precisely, the EPL-1.0 has to be taken as a license with weak copyleft (→ OSLiC, p. 31). The other conditions refer to the distribution in general—no matter what form or state is used.[544] So, taken as whole, the EPL-1.0 mainly focusses on the distribution of software. Thus, for finding the relevant, easy to process task lists, the following EPL-1.0 specific open source use case structure[545] can be used:

[540] cf. *Open Source Initiative*: The CDDL-1.0, 2004, wp. §3.
[541] cf. *Open Source Initiative*: EPL-1.0, 2005, wp §3.
[542] cf. id., l.c., wp §3 top area.
[543] cf. id., l.c., wp §3 mid area.
[544] cf. id., l.c., wp §3 bottom area.
[545] For details of the general OSUC finder → OSLiC, pp. 104 and **??**



6.6.1 EPL-1.0-C1: Using the software only for yourself

means that you received EPL-1.0 licensed software, that you will use it only for yourself, and that you do not hand it over to any 3rd party in any sense.

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L, and OSUC-09N[546]

requires no tasks in order to fulfill the conditions of the Eclipse Public License 1.0 with respect to this use case:

- You are allowed to use any kind of EPL-1.0 software in any sense and in any context without being obliged to do anything as long as you do not give the software to third parties.

prohibits ...

- to remove or to alter any copyright notices that were contained in the software package when you received it.

[546] For details → OSLiC, pp. 112 – 124

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.6.2 EPL-1.0-C2: Passing the unmodified software as source code

means that you received EPL-1.0 licensed software which you are now going to distribute to third parties in the form of unmodified source code files or as unmodified source code package. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or as an embedded unit.

covers OSUC-02S, OSUC-05S, OSUC-07S[547]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly all copyright notices and the disclaimer of warranty and disclaimer of liability) are retained in your package in exactly the form you have received them.
- [**mandatory:**] Give the recipient a copy of the EPL-1.0 license. If it is not already part of the software package, add it. If the licensing statement in the licensing file of the package does still not clearly state that the package is licensed under the EPL-1.0, additionally insert your own correct EPL-1.0 licensing file.[548]
- [**mandatory:**] If still not existing, integrate an explicit, very prominently placed 'No warranty' statement into the distributed source code package. Let this statement clearly say that all (other) contributors to the software do not accept any responsibility for the quality of the software. Then, copy the no-warranty clause and the disclaimer of liability from the EPL-1.0 itself into that file.
- [**voluntary:**] Let the documentation of your distribution or your additional material reproduce the content of an existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EPL-1.0 license.

prohibits ...

- to remove or to alter any copyright notices that were contained in the software package when you received it.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

[547] For details → OSLiC, pp. 112 – 120
[548] For implementing the handover of files correctly → OSLiC, p. 127

6.6.3 EPL-1.0-C3: Passing the unmodified software as binaries

means that you received EPL-1.0 licensed software which you are now going to distribute to third parties in the form of unmodified binary files or as unmodified binary package. In this case it does not matter if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit.

covers OSUC-02B, OSUC-05B, OSUC-07B[549]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly all copyright notices and the disclaimer of warranty and disclaimer of liability) are retained in your package in exactly the form you have received them. If you compile the binary from the sources, ensure that all these licensing elements are also incorporated into the package.
- [**mandatory:**] If still not existing, integrate an explicit, very prominently placed 'No warranty' statement into the distributed binary package. Let this statement clearly say that all (other) contributors to the software do not accept any responsibility for the quality of the software. Then, copy the no-warranty clause and the disclaimer of liability from the EPL-1.0 itself into that file.
- [**mandatory:**] Make the source code of the software accessible through a repository under your own control, even if you did not modify it: Push the source code package into an internet repository and enable the download function. Ensure that this respository is available for a reasonable period of time.
- [**mandatory:**] Insert a prominent hint to the download repository into your distribution or your additional material and explain how the code can be obtained.
- [**mandatory:**] Execute the to-do list of use case EPL-1.0-C2 for the source code that you publish.[550]
- [**voluntary:**] Let the documentation of your distribution or your additional material reproduce the content of an existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EPL-1.0 license.

prohibits ...

[549] For details → OSLiC, pp. 113 – 121

[550] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

- to remove or to alter any copyright notices that were contained in the software package when you received it.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.6.4 EPL-1.0-C4: Passing a modified program as source code

means that you received an EPL-1.0 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

covers OSUC-04S[551]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly all copyright notices and the disclaimer of warranty and disclaimer of liability) are retained in your package in exactly the form you have received them.
- [**mandatory:**] Create a *modification text file* if such a file does not exist. *Add* a general description of your modifications to the *modification text file*. Incorporate it into your distribution package.
- [**mandatory:**] Mark all modifications of the source code of the program thoroughly; namely within the modified source code.
- [**mandatory:**] Give the recipient a copy of the EPL-1.0 license. If it is not already part of the software package, add it. If the licensing statement in the licensing file of the package does still not clearly state that the package is licensed under the EPL-1.0, additionally insert your own correct EPL-1.0 licensing file.[552]
- [**mandatory:**] Organize your modifications in a way that they are covered by the existing EPL-1.0 licensing statements. If you add new source code files, insert a header containing your copyright line and an EPL-1.0 adequate licensing the statement.
- [**mandatory:**] If still not existing, integrate an explicit, very prominently placed 'No warranty' statement into the distributed source code package. Let this statement clearly say that all (other) contributors to the software do not accept any responsibility for the quality of the software. Then, copy the no-warranty clause and the disclaimer of

[551] For details → OSLiC, pp. 116
[552] For implementing the handover of files correctly → OSLiC, p. 127

liability from the EPL-1.0 itself into that file. Update an existing copyright screen presented by the program so that it shows the same information.

- [**voluntary:**] Let the documentation of your distribution or your additional material reproduce the content of an existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EPL-1.0 license.

prohibits ...

- to remove or to alter any copyright notices that were contained in the software package when you received it.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.6.5 EPL-1.0-C5: Passing a modified program as binary

means that you received an EPL-1.0 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package.

covers OSUC-04B[553]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly all copyright notices and the disclaimer of warranty and disclaimer of liability) are retained in your package in exactly the form you have received them. If you compile the binary from the sources, ensure that all these licensing elements are also incorporated into the package.
- [**mandatory:**] Create a *modification text file* if such a file does not exist. *Add* a general description of your modifications to the *modification text file*. Incorporate it into your distribution package.
- [**mandatory:**] Mark all modifications of the source code of the program thoroughly; namely within the modified source code.
- [**mandatory:**] Organize your modifications in a way that they are covered by the existing EPL-1.0 licensing statements.
- [**mandatory:**] If still not existing, integrate an explicit, very prominently placed 'No warranty' statement into the distributed binary package. Let this statement clearly say that all (other) contributors

[553] For details → OSLiC, pp. 116

to the software do not accept any responsibility for the quality of the software. Then, copy the no-warranty clause and the disclaimer of liability from the EPL-1.0 itself into that file. Update an existing copyright screen presented by the program so that it shows the same information.

- [**mandatory:**] Make the source code of the program accessible through a repository under your own control: Push the source code package into an internet repository and enable the download function. Ensure that this respository is available for a reasonable period of time.
- [**mandatory:**] Insert a prominent hint to the download repository into your distribution or your additional material and explain how the code can be obtained.
- [**mandatory:**] Execute the to-do list of use case EPL-1.0-C4 for the source code that you publish.[554]
- [**voluntary:**] Let the documentation of your distribution or your additional material reproduce the content of an existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EPL-1.0 license, preferably as a subsection of your own copyright notice.

prohibits ...

- to remove or to alter any copyright notices that were contained in the software package when you received it.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.6.6 EPL-1.0-C6: Passing a modified library as independent source code

means that you received an EPL-1.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

covers OSUC-08S[555]

requires the following tasks in order to fulfill the license conditions:

[554] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

[555] For details → OSLiC, pp. 122

- [**mandatory:**] Ensure that the licensing elements (particularly all copyright notices and the disclaimer of warranty and disclaimer of liability) are retained in your package in exactly the form you have received them.
- [**mandatory:**] Create a *modification text file* if such a file does not exist. *Add* a general description of your modifications to the *modification text file*. Incorporate it into your distribution package.
- [**mandatory:**] Mark all modifications of the source code of the program thoroughly; namely within the modified source code.
- [**mandatory:**] Give the recipient a copy of the EPL-1.0 license. If it is not already part of the software package, add it. If the licensing statement in the licensing file of the package does still not clearly state that the package is licensed under the EPL-1.0, additionally insert your own correct EPL-1.0 licensing file.[556]
- [**mandatory:**] Organize your modifications in a way that they are covered by the existing EPL-1.0 licensing statements. If you add new source code files, insert a header containing your copyright line and an EPL-1.0 adequate licensing the statement.
- [**mandatory:**] If still not existing, integrate an explicit, very prominently placed 'No warranty' statement into the distributed source code package. Let this statement clearly say that all (other) contributors to the software do not accept any responsibility for the quality of the software. Then, copy the no-warranty clause and the disclaimer of liability from the EPL-1.0 itself into that file.
- [**voluntary:**] Let the documentation of your distribution or your additional material reproduce the content of an existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EPL-1.0 license.

prohibits ...

- to remove or to alter any copyright notices that were contained in the software package when you received it.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

[556] For implementing the handover of files correctly → OSLiC, p. 127

6.6.7 EPL-1.0-C7: Passing a modified library as independent binary

means that you received an EPL-1.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers OSUC-08B[557]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly all copyright notices and the disclaimer of warranty and disclaimer of liability) are retained in your package in exactly the form you have received them. If you compile the binary from the sources, ensure that all these licensing elements are also incorporated into the package.
- [**mandatory:**] Create a *modification text file* if such a file does not exist. *Add* a general description of your modifications to the *modification text file*. Incorporate it into your distribution package.
- [**mandatory:**] Mark all modifications of the source code of the program thoroughly; namely within the modified source code.
- [**mandatory:**] Organize your modifications in a way that they are covered by the existing EPL-1.0 licensing statements.
- [**mandatory:**] If still not existing, integrate an explicit, very prominently placed 'No warranty' statement into the distributed binary package. Let this statement clearly say that all (other) contributors to the software do not accept any responsibility for the quality of the software. Then, copy the no-warranty clause and the disclaimer of liability from the EPL-1.0 itself into that file.
- [**mandatory:**] Make the source code of the modified library accessible through a repository under your own control: Push the source code package into an internet repository and enable the download function. Ensure that this respository is available for a reasonable period of time.
- [**mandatory:**] Insert a prominent hint to the download repository into your distribution or your additional material and explain how the code can be obtained.
- [**mandatory:**] Execute the to-do list of use case EPL-1.0-C6 for the source code that you publish.[558]

[557] For details → OSLiC, pp. 123

[558] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

- [**voluntary:**] Let the documentation of your distribution or your additional material reproduce the content of an existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EPL-1.0 license.

prohibits ...

- to remove or to alter any copyright notices that were contained in the software package when you received it.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.6.8 EPL-1.0-C8: Passing a modified library as embedded source code

means that you received an EPL-1.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10S[559]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly all copyright notices and the disclaimer of warranty and disclaimer of liability) are retained in your package in exactly the form you have received them.
- [**mandatory:**] Create a *modification text file* if such a file does not exist. *Add* a general description of your modifications to the *modification text file*. Incorporate it into your distribution package.
- [**mandatory:**] Mark all modifications of the source code of the program thoroughly; namely within the modified source code.
- [**mandatory:**] Give the recipient a copy of the EPL-1.0 license. If it is not already part of the software package, add it. If the licensing statement in the licensing file of the package does still not clearly state that the package is licensed under the EPL-1.0, additionally insert your own correct EPL-1.0 licensing file.[560]
- [**mandatory:**] If still not existing, integrate an explicit, very prominently placed 'No warranty' statement into the distributed source code

[559] For details → OSLiC, pp. 125
[560] For implementing the handover of files correctly → OSLiC, p. 127

package. Let this statement clearly say that all (other) contributors to the software do not accept any responsibility for the quality of the software. Then, copy the no-warranty clause and the disclaimer of liability from the EPL-1.0 itself into that file. Let the copyright screen of your own overarching program show the same information as a specification for the embedded component.

- [**mandatory:**] Organize your modifications in a way that they are covered by the existing EPL-1.0 licensing statements. If you add new source code files, insert a header containing your copyright line and an EPL-1.0 adequate licensing the statement.
- [**voluntary:**] Arrange your source code distribution so that the integrated EPL-1.0 and the *licensing files* 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 separate directories which also contains all additional licensing elements.
- [**voluntary:**] Let the documentation of your distribution or your additional material reproduce the content of an existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EPL-1.0 license, preferably as a subsection of your own copyright notice.

prohibits ...

- to remove or to alter any copyright notices that were contained in the software package when you received it.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.6.9 EPL-1.0-C9: Passing a modified library as embedded binary

means that you received an EPL-1.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10B[561]

requires the following tasks in order to fulfill the license conditions:

[561] For details → OSLiC, pp. 126

- [**mandatory:**] Ensure that the licensing elements (particularly all copyright notices and the disclaimer of warranty and disclaimer of liability) are retained in your package in exactly the form you have received them. If you compile the binary from the sources, ensure that all these licensing elements are also incorporated into the package.
- [**mandatory:**] Create a *modification text file* if such a file does not exist. *Add* a general description of your modifications to the *modification text file*. Incorporate it into your distribution package.
- [**mandatory:**] Mark all modifications of the source code of the program thoroughly; namely within the modified source code.
- [**mandatory:**] If still not existing, integrate an explicit, very prominently placed 'No warranty' statement into the distributed binary package. Let this statement clearly say that all (other) contributors to the software do not accept any responsibility for the quality of the software. Then, copy the no-warranty clause and the disclaimer of liability from the EPL-1.0 itself into that file. Let the copyright screen of your own overarching program show the same information as a specification for the embedded component.
- [**mandatory:**] Make the source code of the embedded library accessible through a repository under your own control: Push the source code package into an internet repository and enable the download function. Ensure that this respository is available for a reasonable period of time.
- [**mandatory:**] Insert a prominent hint to the download repository into your distribution or your additional material and explain how the code can be obtained.
- [**mandatory:**] Organize your modifications in a way that they are covered by the existing EPL-1.0 licensing statements.
- [**mandatory:**] Execute the to-do list of use case EPL-1.0-C8 for the source code that you publish.[562]
- [**voluntary:**] Arrange your binary distribution so that the integrated EPL-1.0 and the *licensing files* 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 separate directories which also contains all additional licensing elements.
- [**voluntary:**] Let the documentation of your distribution or your additional material reproduce the content of an existing *copyright*

[562] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

notice text files, a hint to the software name, a link to its homepage, and a link to the EPL-1.0 license, preferably as a subsection of your own copyright notice.

prohibits ...

- to remove or to alter any copyright notices that were contained in the software package when you received it.
- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.6.10 Discussions and Explanations

The EPL-1.0 contains a succinct section "Requirements"[563] complemented by some definitions concerning a "Commercial Distribution"[564]: First, it describes what a distributor must do for correctly distributing an Eclipse licensed program as a set of binaries. Then, it explains, what must be done to comply with the license when distributing the software as source code. Finally, it lists two conditions which must be fulfilled in any case.[565] With respect to this structure, we can discover the following tasks:

- The EPL-1.0 generally requires that "Contributors may not remove or alter any copyright notices contained within the Program"[566] where the word 'Contributor' has to be read as "any person or entity that distributes the Program", and the word 'Program' denotes the "initial contribution" and all its modifications.[567] Similar to the EUPL and at least in a very strict reading, the EPL-1.0 does not limit these requirements to the distribution of the software (*2others*). But in practice it will be difficult to control the compliant use of the software in those cases where one uses the software only for oneself. But opposite to, for example, the EUPL, the EPL-1.0 clearly contains this interdiction. The OSLiC solves this practical inconsistence duplicating the message: On the one hand, it rewrites the negative condition as a mandatory positive assertion for the *2others* use cases (EPL-1.0-C2 – EPL-1.0-C9). This should emphasize the *activity* to retain the copyright notes in exact the form one has received them. On the other hand, the OSLiC inserts the corresponding interdiction into the 'prohibits' section of the *4yourself* use cases (EPL-1.0-C1 – EPL-1.0-C9).

563) cf. *Open Source Initiative*: EPL-1.0, 2005, wp §3.
564) cf. id., l.c., wp §4.
565) cf. id., l.c., wp §3.
566) cf. id., ibid.
567) cf. id., l.c., wp §1.

- Furthermore, the EPL-1.0 requires that "each Contributor must identify itself as the originator of its Contributions [...] in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution",[568] In this case, 'Contribution' has to be read as the "initial code and documention" together with all subsequent modifications of these parts.[569] To fulfill this condition faithfully, a developer must mark and describe his modifications of the source code within this source code; and the distributor must describe these modifications on the more general level of software features in a file sometimes called CHANGES. At a first glance, the requirement to document the source code modifications within the source code seems to be restricted to the use cases which concern the distribution of a modified EPL-1.0 software in the form of source code. But the EPL-1.0 allows the distribution in the form of binaries only if the distributor also states where one can obtain the correspoding code.[570] So, distributing the binaries implies the distribution of the source code. Therefore the OSLiC inserts the two requirements as mandatory clauses into all the use cases concerning the distribution of a modified EPL-1.0 software (EPL-1.0-C4 – EPL-1.0-C9).
- For all distributions in the form of source code the EPL-1.0 requires that the software "[...] must be made available under this (Eclipse Public License 1.0) Agreement" and that "[...] a copy of this Agreement must be included with each copy of the Program."[571] Thus, the OSLiC inserts a respective mandatory clause into the use cases (EPL-1.0-C4, EPL-1.0-C6, EPL-1.0-C8). But the EPL-1.0 is a license with a weak copyleft[572]. Therefore, this conditions does not cover the overarching program which uses the embedded library (EPL-1.0-C8).
- Additionally, the EPL-1.0 allows to distribute the software in the form of binaries if the distributor "[...] effectively disclaims on behalf of all Contributors all warranties and conditions [...] (and) effectively excludes on behalf of all Contributors all liability for damages [...]" in the broadest sense.[573] This limitation of liability is very important to the EPL-1.0. Thus, it further specifies and explains this aspect once more in another section titled "Commercial Distribution". There, this aspect is no longer focussed only on a distribution in the form of binaries.[574] So the OSLiC inserts a mandatory clause into all use cases concerning the distribution that the

568) cf. *Open Source Initiative*: EPL-1.0, 2005, wp §3.
569) cf. id., l.c., wp §1.
570) cf. id., l.c., wp §3.
571) cf. id., ibid.
572) (→ OSLiC, p. 31)
573) cf. id., ibid.
574) cf. id., l.c., wp §4.

paragraph of "No Warranty"[575] and the "Disclaimer of Liability"[576] of the EPL-1.0 must explicitly be present in the documentation of distribution package and—if technically possible—presented by the copyright screen.

- Aside from that, the EPL-1.0 allows the distribution of the software in the form of binaries only if the distributor clearly "[...] states that the source code for the program is available from such Contributor (distributor) [...]" and if he additionally "[...] informs licensees how to obtain it in a reasonable manner [...]"[577] This requirement can only be fulfilled seriously if the distributor himself offers the source code via a repository. It is not sufficient to point to any external download repository in the world wide web. Thus,—for all use cases concerning the distribution in the form of binaries—the OSLiC follows the respective requirement introduced by the EPL-1.0 (EPL-1.0-C3, EPL-1.0-C5, EPL-1.0-C7, EPL-1.0-C9).
- Moreover, one has clearly to state that the previous rule implies a real source code distribution which therefore must follow the rules of distributing the software. Thus, the OSLiC requires in all cases of a binary distribution to execute also the task-lists of the respective source code use cases.
- Finally, the EPL-1.0 contains a patent clause stating that "if any recipient institutes patent litigation against any entity [...] alleging that the Program itself [...] infringes such Recipient's patent(s), then such Recipient's rights granted [... by the EPL-1.0] shall terminate [...]"[578]. Based on this fact, the OSLiC generally (EPL-1.0-C1 – EPL-1.0-C9) interdicts to legally fight against patents linked to the software.

6.7 EUPL-1.1 licensed software

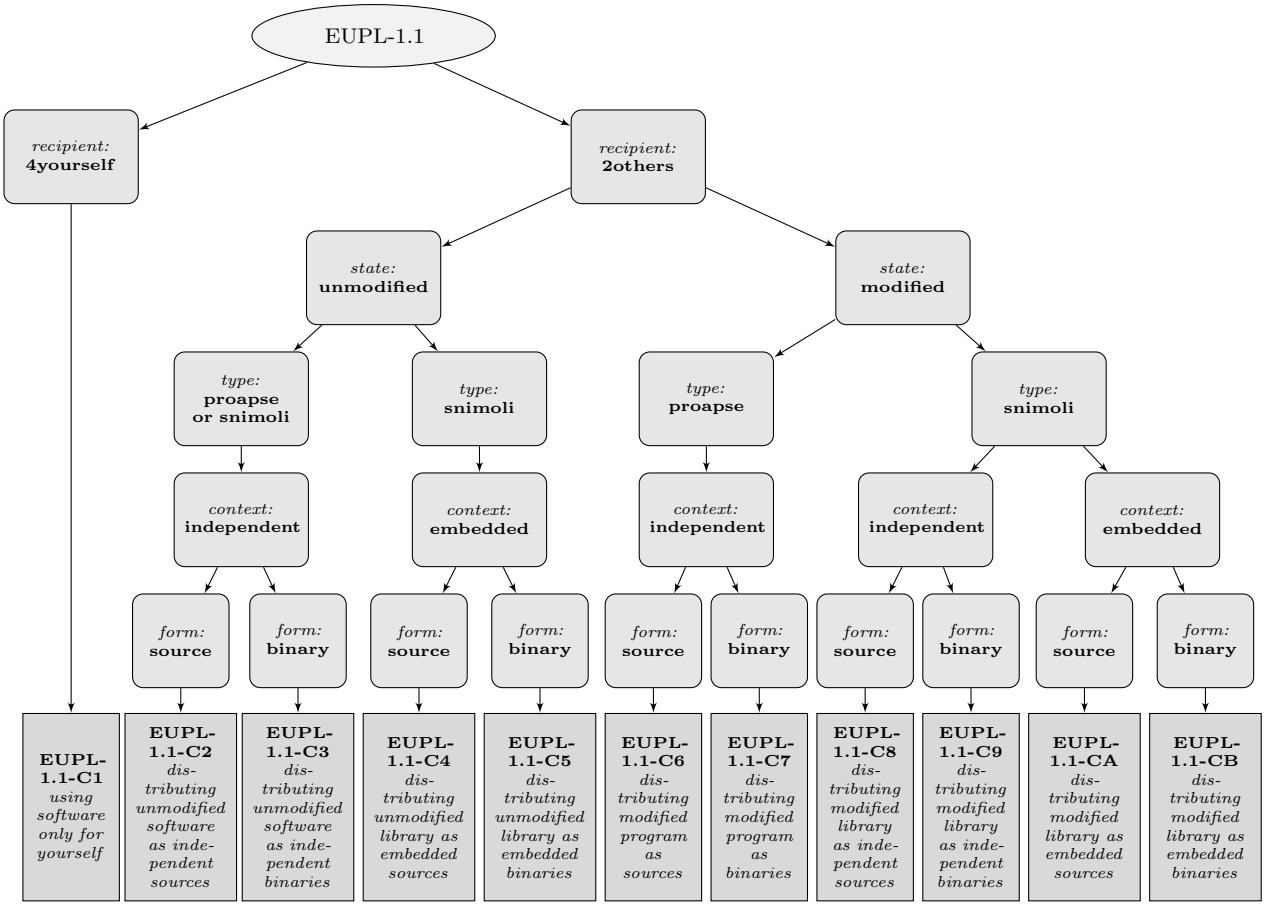
The European Union Public License explicitly distinguishes the distribution of the source code from that of the binaries: In the chapter "Communication of the Source Code," it allows to "provide the Work either in its Source Code form, or as Executable Code."[579] But if a piece of EUPL-1.1 licensed software is distributed as binary package, then the license additionally requires that the distributor either "[...] provides a machine-readable copy of the Source Code [...]" directly together with the binaries[580] or that he "[...] indicates [...] a repository where the Source Code is easily and freely accessible for as long as the Licensor continues

575) cf. *Open Source Initiative*: EPL-1.0, 2005, wp §5.
576) cf. id., l.c., wp §6.
577) cf. id., l.c., wp §3.
578) cf. id., l.c., wp §7.
579) cf. *Open Source Initiative*: EUPL-1.1 (OSI), 2007, wp §3.
580) cf. id., l.c., wp §5.

to distribute [...] the Work."[581] For respecting this conditions it is irrelevant whether the software has been modified or not and all the other "obligations of the licensee" refer to both forms.[582]

There is a particular aspect which has to be considered for acting in accordance to the EUPL-1.1: Taken literally, the EUPL is a license with a weak copyleft, no doubt. But this happens only a result of the fact that the EUPL-1.1 allows the licensee to relicense the software by following the conditions of the "Compatibility clause"[583] and an license listed in an appendix, which also includes some licenses with a weak copyleft.[584] But, with respect to question how to fulfill the license best, it is safer to treat the EUPL-1.1 as a license with a strong copyleft. Concerning the use of an unmodified or a modified library as an embedded component, a license with a strong copyleft implies that the application which is using the (un)modified library has also to be licensed under the same conditions as the library itself. Thus, to find a simple to process task lists, use the following EUPL-1.1 specific open source use case structure:[585]

[581] cf. *Open Source Initiative*: EUPL-1.1 (OSI), 2007, wp §3.
[582] cf. id., l.c., wp §5.
[583] cf. id., ibid.
[584] (→ OSLiC, p. 33)
[585] For details of the general OSUC finder → OSLiC, pp. 104 and **??**



6.7.1 EUPL-1.1-C1: Using the software only for yourself

means that you received EUPL-1.1 licensed software, that you will use it only for yourself and that you do not hand it over to any 3rd party in any sense.

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L, and OSUC-09N[586]

requires no tasks in order to fulfill the conditions of the European Union Public License 1.1 with respect to this use case:

- You are allowed to use any kind of EUPL-1.1 software in any sense and in any context without being obliged to do anything as long as you do not give the software to third parties.

prohibits ...

- to promote any of your services or products based on the this software by trade names, trademarks, service marks, or names linked to this

[586] For details → OSLiC, pp. 112 - 124

EUPL-1.1 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.7.2 EUPL-1.1-C2: Passing the unmodified software as independent sources

means that you received EUPL-1.1 licensed software which you are now going to distribute to third parties as an independent unit and in the form of unmodified source code files or as unmodified source code package. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or as an embedded unit.

covers OSUC-02S, OSUC-05S[587]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly the copyright, patent, and trademark notices and all notices that refer to the license or to the disclaimer of warranties) are retained in your package in the form you have received them.
- [**mandatory:**] Give the recipient a copy of the EUPL-1.1 license. If it is not already part of the software package, add it.[588]
- [**voluntary:**] Let the documentation of your distribution or your additional material also reproduce the content of the existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EUPL-1.1 license.

prohibits ...

- to promote any of your services or products based on the this software by trade names, trademarks, service marks, or names linked to this EUPL-1.1 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.7.3 EUPL-1.1-C3: Passing the unmodified software as independent binaries

means that you received EUPL-1.1 licensed software which you are now going to distribute to third parties as an independent unit and in the form of

[587] For details → OSLiC, pp. 112 - 117
[588] For implementing the handover of files correctly → OSLiC, p. 127

unmodified binary files or as unmodified binary package. In this case it does not matter if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit.

covers OSUC-02B, OSUC-05B[589]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly the copyright, patent, and trademark notices and all notices that refer to the license or to the disclaimer of warranties) 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:**] Give the recipient a copy of the EUPL-1.1 license. If it is not already part of the software package, add it.[590]
- [**mandatory:**] Make the source code of the distributed software accessible via a repository under your own control (even if you did not modify it): Push the source code package into a repository, make it downloadable via the internet, and include an easy to find description in the distribution package, which explains how and where the code can be received. Ensure, that this repository is online for as long as you continue to distribute the software.
- [**mandatory:**] Insert a prominent hint to the download repository into your distribution or your additional material.
- [**mandatory:**] Execute the to-do list of use case EUPL-1.1-C2 for the source code that you publish.[591]
- [**voluntary:**] Let the documentation of your distribution or your additional material also reproduce the content of the existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EUPL-1.1 license.

prohibits ...

- to promote any of your services or products based on the this software by trade names, trademarks, service marks, or names linked to this EUPL-1.1 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

589) For details → OSLiC, pp. 113 - 118

590) For implementing the handover of files correctly → OSLiC, p. 127

591) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

6.7.4 EUPL-1.1-C4: Passing the unmodified library as embedded sources

means that you received a EUPL-1.1 licensed snippet, module or library which you are now going to distribute to third parties as an embedded component of a larger unit and in the form of unmodified source code files or as unmodified source code package.

covers OSUC-07S[592]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly the copyright, patent, and trademark notices and all notices that refer to the license or to the disclaimer of warranties) are retained in your package in the form you have received them.
- [**mandatory:**] Give the recipient a copy of the EUPL-1.1 license. If it is not already part of the software package, add it.[593]
- [**mandatory:**] License your program, which includes the library, also under the EUPL-1.1. Arrange the sources of the on-top development in a way that they are also covered by the EUPL-1.1 licensing statements.
- [**voluntary:**] Let the copyright dialog of the on-top development clearly say, that it uses the EUPL-1.1 licensed library and that it is itself licensed under the EUPL-1.1, too.
- [**voluntary:**] Let the documentation of your distribution or your additional material also reproduce the content of the existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EUPL-1.1 license, preferably as a subsection of your own copyright notice.

prohibits ...

- to promote any of your services or products based on the this software by trade names, trademarks, service marks, or names linked to this EUPL-1.1 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.7.5 EUPL-1.1-C5: Passing the unmodified library as embedded binaries

means that you received a EUPL-1.1 licensed snippet, module or library which you are now going to distribute to third parties as an embedded component

[592] For details → OSLiC, pp. 120
[593] For implementing the handover of files correctly → OSLiC, p. 127

of a larger unit and in the form of unmodified binary files or as unmodified binary package.

covers OSUC-07B[594]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly the copyright, patent, and trademark notices and all notices that refer to the license or to the disclaimer of warranties) 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:**] Give the recipient a copy of the EUPL-1.1 license. If it is not already part of the software package, add it.[595]
- [**mandatory:**] Make the source code of the embedded library *and* your overarching program accessible via a repository under your own control (even if you did not modify it): Push the source code package into a repository, make it downloadable via the internet, and include an easy to find description in the distribution package, which explains how and where the code can be received. Ensure, that this repository is online for as long as you continue to distribute the software.
- [**mandatory:**] Insert a prominent hint to the download repository into your distribution or your additional material.
- [**mandatory:**] License your program, which includes the library, also under the EUPL-1.1. Arrange the binaries of the on-top development in a way that they are also covered by the EUPL-1.1 licensing statements.
- [**mandatory:**] Execute the to-do list of use case EUPL-1.1-C4 for the source code that you publish.[596]
- [**voluntary:**] Let the copyright dialog of the on-top development clearly say, that it uses the EUPL-1.1 licensed library and that it is itself licensed under the EUPL-1.1, too.
- [**voluntary:**] Let the documentation of your distribution or your additional material also reproduce the content of the existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EUPL-1.1 license, preferably as a subsection of your own copyright notice.

[594] For details → OSLiC, pp. 121

[595] For implementing the handover of files correctly → OSLiC, p. 127

[596] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

prohibits ...

- to promote any of your services or products based on the this software by trade names, trademarks, service marks, or names linked to this EUPL-1.1 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.7.6 EUPL-1.1-C6: Passing a modified program as source code

means that you received a EUPL-1.1 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

covers OSUC-04S[597]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly the copyright, patent, and trademark notices and all notices that refer to the license or to the disclaimer of warranties) are retained in your package in the form you have received them.
- [**mandatory:**] Give the recipient a copy of the EUPL-1.1 license. If it is not already part of the software package, add it.[598]
- [**mandatory:**] Create a *modification text file*, if such a file still does not exist. *Add* a description of your modifications to the *modification text file.*
- [**mandatory:**] Mark all modifications of source code of the program thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications in a way that they are covered by the existing EUPL-1.1 licensing statements. If you add new source code files, insert a header containing your copyright line and an EUPL-1.1 adequate licensing the statement.
- [**voluntary:**] Let the documentation of your distribution or your additional material also reproduce the content of the existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EUPL-1.1 license.

prohibits ...

[597] For details → OSLiC, pp. 116
[598] For implementing the handover of files correctly → OSLiC, p. 127

- to promote any of your services or products based on the this software by trade names, trademarks, service marks, or names linked to this EUPL-1.1 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.7.7 EUPL-1.1-C7: Passing a modified program as binary

means that you received a EUPL-1.1 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package.

covers OSUC-04B[599]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly the copyright, patent, and trademark notices and all notices that refer to the license or to the disclaimer of warranties) 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:**] Give the recipient a copy of the EUPL-1.1 license. If it is not already part of the software package, add it.[600]
- [**mandatory:**] Create a *modification text file*, if such a file still does not exist. *Add* a description of your modifications to the *modification text file.*
- [**mandatory:**] Arrange your modifications in a way that they are covered by the existing EUPL-1.1 licensing statements.
- [**mandatory:**] Make the source code of the distributed software accessible via a repository under your own control (even if you did not modify it): Push the source code package into a repository, make it downloadable via the internet, and include an easy to find description in the distribution package, which explains how and where the code can be received. Ensure, that this repository is online for as long as you continue to distribute the software.
- [**mandatory:**] Insert a prominent hint to the download repository into your distribution or your additional material.

[599] For details → OSLiC, pp. 116
[600] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Execute the to-do list of use case EUPL-1.1-C6 for the source code that you publish.[601]
- [**voluntary:**] Mark all modifications of source code of the program thoroughly within the source code and include the date of the modification.
- [**voluntary:**] Let the documentation of your distribution or your additional material also reproduce the content of the existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EUPL-1.1 license.

prohibits ...

- to promote any of your services or products based on the this software by trade names, trademarks, service marks, or names linked to this EUPL-1.1 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.7.8 EUPL-1.1-C8: Passing a modified library as independent source code

means that you received a EUPL-1.1 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

covers OSUC-08S[602]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly the copyright, patent, and trademark notices and all notices that refer to the license or to the disclaimer of warranties) are retained in your package in the form you have received them.
- [**mandatory:**] Give the recipient a copy of the EUPL-1.1 license. If it is not already part of the software package, add it.[603]
- [**mandatory:**] Create a *modification text file*, if such a file still does not exist. *Add* a description of your modifications to the *modification text file.*

[601] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.
[602] For details → OSLiC, pp. 122
[603] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Mark all modifications of source code of the library thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications in a way that they are covered by the existing EUPL-1.1 licensing statements. If you add new source code files, insert a header containing your copyright line and an EUPL-1.1 adequate licensing the statement.
- [**voluntary:**] Let the documentation of your distribution or your additional material also reproduce the content of the existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EUPL-1.1 license.

prohibits ...

- to promote any of your services or products based on the this software by trade names, trademarks, service marks, or names linked to this EUPL-1.1 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.7.9 EUPL-1.1-C9: Passing a modified library as independent binary

means that you received a EUPL-1.1 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers OSUC-08B[604]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly the copyright, patent, and trademark notices and all notices that refer to the license or to the disclaimer of warranties) 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:**] Give the recipient a copy of the EUPL-1.1 license. If it is not already part of the software package, add it.[605]

[604] For details → OSLiC, pp. 123
[605] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Create a *modification text file*, if such a file still does not exist. *Add* a description of your modifications to the *modification text file.*
- [**mandatory:**] Arrange your modifications in a way that they are covered by the existing EUPL-1.1 licensing statements.
- [**mandatory:**] Make the source code of the distributed software accessible via a repository under your own control (even if you did not modify it): Push the source code package into a repository, make it downloadable via the internet, and include an easy to find description in the distribution package, which explains how and where the code can be received. Ensure, that this repository is online for as long as you continue to distribute the software.
- [**mandatory:**] Insert a prominent hint to the download repository into your distribution or your additional material.
- [**mandatory:**] Execute the to-do list of use case EUPL-1.1-C8 for the source code that you publish.[606]
- [**voluntary:**] Mark all modifications of source code of the library thoroughly within the source code and include the date of the modification.
- [**voluntary:**] Let the documentation of your distribution or your additional material also reproduce the content of the existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EUPL-1.1 license.

prohibits ...

- to promote any of your services or products based on the this software by trade names, trademarks, service marks, or names linked to this EUPL-1.1 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.7.10 EUPL-1.1-CA: Passing a modified library as embedded source code

means that you received a EUPL-1.1 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

606) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

covers OSUC-10S[607]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly the copyright, patent, and trademark notices and all notices that refer to the license or to the disclaimer of warranties) are retained in your package in the form you have received them.
- [**mandatory:**] Give the recipient a copy of the EUPL-1.1 license. If it is not already part of the software package, add it.[608]
- [**mandatory:**] Create a *modification text file*, if such a file still does not exist. *Add* a description of your modifications to the *modification text file.*
- [**mandatory:**] Arrange your modifications in a way that they are covered by the existing EUPL-1.1 licensing statements. If you add new source code files, insert a header containing your copyright line and an EUPL-1.1 adequate licensing the statement.
- [**mandatory:**] License your program, which includes the library, also under the EUPL-1.1. Arrange the sources of the on-top development in a way that they are also covered by the EUPL-1.1 licensing statements.
- [**mandatory:**] Mark all modifications of source code of the embedded library thoroughly within the source code and include the date of the modification.
- [**voluntary:**] Let the copyright dialog of the on-top development clearly say, that it uses the EUPL-1.1 licensed library and that it is itself licensed under the EUPL-1.1, too.
- [**voluntary:**] Let the documentation of your distribution or your additional material also reproduce the content of the existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EUPL-1.1 license, preferably as a subsection of your own copyright notice.

prohibits ...

- to promote any of your services or products based on the this software by trade names, trademarks, service marks, or names linked to this EUPL-1.1 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

607) For details → OSLiC, pp. 125
608) For implementing the handover of files correctly → OSLiC, p. 127

6.7.11 EUPL-1.1-CB: Passing a modified library as embedded binary

means that you received a EUPL-1.1 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10B[609]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (particularly the copyright, patent, and trademark notices and all notices that refer to the license or to the disclaimer of warranties) 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:**] Give the recipient a copy of the EUPL-1.1 license. If it is not already part of the software package, add it.[610]
- [**mandatory:**] Create a *modification text file*, if such a file still does not exist. *Add* a description of your modifications to the *modification text file.*
- [**mandatory:**] Make the source code of the embedded library *and* your overarching program accessible via a repository under your own control (even if you did not modify it): Push the source code package into a repository, make it downloadable via the internet, and include an easy to find description in the distribution package, which explains how and where the code can be received. Ensure, that this repository is online for as long as you continue to distribute the software.
- [**mandatory:**] Insert a prominent hint to the download repository into your distribution or your additional material.
- [**mandatory:**] Execute the to-do list of use case EUPL-1.1-CA for the source code that you publish.[611]
- [**mandatory:**] Arrange your modifications in a way that they are covered by the existing EUPL-1.1 licensing statements.
- [**mandatory:**] License your program, which includes the library, also under the EUPL-1.1. Arrange the binaries of the on-top development in

[609] For details → OSLiC, pp. 126

[610] For implementing the handover of files correctly → OSLiC, p. 127

[611] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

a way that they are also covered by the EUPL-1.1 licensing statements.

- [**voluntary:**] Mark all modifications of source code of the embedded library thoroughly within the source code and include the date of the modification.
- [**voluntary:**] Let the documentation of your distribution or your additional material also reproduce the content of the existing *copyright notice text files*, a hint to the software name, a link to its homepage, and a link to the EUPL-1.1 license, preferably as a subsection of your own copyright notice.

prohibits ...

- to promote any of your services or products based on the this software by trade names, trademarks, service marks, or names linked to this EUPL-1.1 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.7.12 Discussions and Explanations

- The EUPL-1.1 generally "[...] does not grant permission to use the trade names, trademarks, service marks, or names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the copyright notice."[612] Therefore, the OSLiC genreally interdicts (EUPL-1.1-C1 – EUPL-1.1-CB) to promote any service or product based on this software by such elements.
- The EUPL-1.1 generally requires that "[...] the Licensee shall keep intact all copyright, patent or trademarks notices and all notices that refer to the Licence and to the disclaimer of warranties."[613] In a very strict reading, the EUPL-1.1 does not limit this requirement to the distribution of the software. But in practise, it will be impossible to control the compliant use of the software in those cases (*4yourself*) unless you also start to distribute the software. Therefore the OSLiC only inserts this requirement as a mandatory clause only for the *2others* use cases (EUPL-1.1-C2 – EUPL-1.1-CB).
- The EUPL-1.1 also requires to "[...] include [...] a copy of the (EUPL-1.1) Licence with every (distributed) copy of the Work".[614] Therefore, all *2others* use cases contain the respective mandatory condition (EUPL-1.1-C2 – EUPL-1.1-CB).

612) cf. *Open Source Initiative*: EUPL-1.1 (OSI), 2007, wp §5.
613) cf. id., ibid.
614) cf. id., ibid.

- Additionally, the EUPL-1.1 requires that the "licensee" who distributes a modified work "[...] must cause any Derivative Work to carry prominent notices stating that the Work has been modified and the date of modification."[615] Thus, the OSLiC integrates the mandatory requirement to generate (update) a respective notice file into all 'modification' use cases and recommends to mark all modifications in the source code (EUPL-1.1-C6 – EUPL-1.1-CB).
- Furthermore, the EUPL-1.1 requires that any distributor of the software "[...] provide a machine-readable copy of the Source Code [...]" by "[...] (indicating) a repository where this Source will be easily and freely available for as long as the Licensee continues to distribute [...] the Work."[616] Therefore the OSLiC inserts a respective requirement into the task list of all cases concerning a binary distribution (EUPL-1.1-C3, EUPL-1.1-C7, EUPL-1.1-C9, and EUPL-1.1-CB)
- Finally, the EUPL-1.1 contains a "copyleft clause" stating that if a "[...] Licensee distributes [...] copies of the Original Works or Derivative Works based upon the Original Work, this Distribution [...] will be done under the terms of this (EUPL-1.1) Licence [...]". In all the use cases which do not concern the use of an embedded component (EUPL-1.1-C2 – EUPL-1.1-C9) this copyleft clause is already fulfilled by either distributing the modified sources themselves or by making them accessible via a repository. In those cases where the licensee distributes an program that uses an embedded EUPL-1.1 licensed component (EUPL-1.1-CA – EUPL-1.1-CB), in general, the code of the embedding program must also be distributed. Thus, with respect to the use case (EUPL-1.1-CA) this is already fulfilled by definition. Therefore, the OSLiC only mentions this default view in the case EUPL-1.1-CB implying a strong copyleft effect.[617]

6.8 GPL licensed software

Both versions of the GNU General Public License explicitly distinguish the distribution of the source code from that of the binaries: On the one hand,

615) cf. *Open Source Initiative*: EUPL-1.1 (OSI), 2007, wp §5.

616) cf. *European Community a. European commission Joinup*: EUPL-1.1/EN, 2007, wp. §5. To be precise, the EUPL-1.1 also allows to directly distribute the source code together with the binary packages (cf. id., l.c., wp. §3). With respect to the OSLiC principle to offer only one reliable way, the OSLiC simplifies this option: It 'only' asks for the repository solution.

617) Formally, the EUPL-1.1 is only a license with weak copyleft. But this is only a result of allowing to relicense the software (→ OSLiC, p. 33). So, as long as you do not relicense the embedded library with respect to the list of "compatible licenses according to article 5 EUPL-1.1" (cf. id., l.c., wp §5 and Appendix), you also have to publish the code of your overarching work.

the GPL-2.0 mainly talks about copying and distributing the source code,[618] but also mentions the specific conditions for "[...] (copying) and (distributing) the Program [...] in object code or executable form [...]"[619] On the other hand, the GPL-3.0 describes the "Basic Permissions" and the conditions for "Conveying Verbatim Copies" or for "Conveying Modified Source Versions"[620] before it explains the rules for "Conveying Non-Source-Forms".[621]

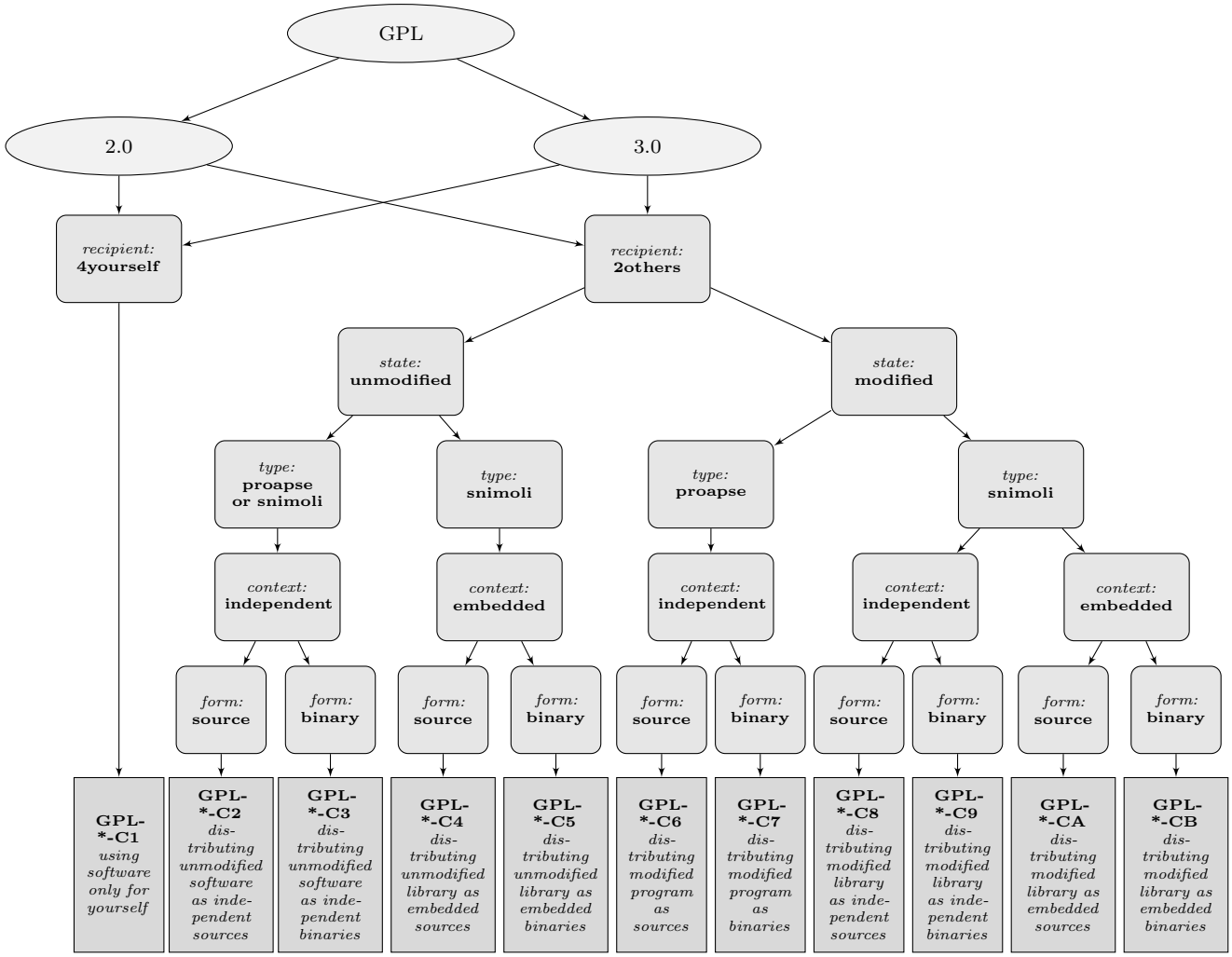
GPL-2.0 and GPL-3.0 mainly talk about copying *and* distributing the software; private use is nearly completely unspecified: The GPL-2.0 lists its 'restrictions' only with respect to the act of copying *and* distributing "copies of the program"[622] while the GPL-3.0 explicitly specifies that one "[...] may make, run and propagate covered works that (one does) not convey, without conditions so long as (the) license otherwise remains in force."[623]

As licenses with a strong copyleft, they require that any application that contains a GPL-licensed library must itself be licensed under the same conditions as the library.

Finally, the GPL-2.0 and the GPL-3.0 aim for the same results and share the same spirit by requiring nearly the same task to be performed for fulfilling the license conditions. Therefore it is appropriate to cover both versions in the same chapter and to offer a common specialized GPL open source use case structure for quickly finding the appropriate task list.[624] However, the task lists themselves will be kept separate.

In the following diagram, GPL-*-C1 (GPL-*-C2, ..., GPL-*-CB) is either GPL-2.0-C1 (and so forth), if you are looking for the GPL-2.0 use case, or GPL-3.0-C1, ... for the GPL-3.0 use case.

618) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp §1, §2.
619) cf. id., l.c., wp §3.
620) cf. id., l.c., wp §2, §4, §5.
621) cf. id., ibid.
622) cf. id., l.c., wp §1, §2, §4 et passim; emphasize by KR.
623) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp §2.
624) For details of the general OSUC finder → OSLiC, pp. 104 and **??**



6.8.1 GPL-2.0-C1: Using the software only for yourself

means that you received GPL-2.0 licensed software, that you will use it only for yourself, and that you do not hand over to any third party in any sense.

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L and OSUC-09N[625]

requires no tasks in order to fulfill the conditions of the General Public License Version 2 with respect to this use case:

- You are allowed to use any kind of GPL-2.0software in any sense and in any context without being obliged to do anything as long as you do not give the software to third parties.

[625] For details → OSLiC, pp. 112 – 124

prohibits nothing explicitely.

6.8.2 GPL-2.0-C2: Passing the unmodified software as independent sources

means that you received GPL-2.0 licensed software that you are now going to distribute to third parties as an independent unit and in the form of unmodified source code files or as an unmodified source code package. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin.

covers OSUC-02S, OSUC-05S[626]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-2.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the GPL-2.0 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the GPL-2.0 license. If it is not already part of the software package, add it.[627]
- [**mandatory:**] Retain all existing copyright notices.
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.

prohibits nothing explicitely.

6.8.3 GPL-2.0-C3: Passing the unmodified software as independent binaries

means that you received GPL-2.0 licensed software, which you are now going to distribute to third parties as an independent unit and in the form of unmodified binary files or as an unmodified binary package. In this case it does not matter if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin.

626) For details → OSLiC, pp. 112 – 117

627) For implementing the handover of files correctly → OSLiC, p. 127

covers OSUC-02B, OSUC-05B[628]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-2.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the GPL-2.0 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the GPL-2.0 license. If it is not already part of the software package, add it.[629]
- [**mandatory:**] Make the source code of the distributed software publicly available (even though you did not modify it): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Execute the to-do list of use case GPL-2.0-C2 for the source code that you publish.[630]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.

prohibits nothing explicitely.

6.8.4 GPL-2.0-C4: Passing the unmodified library as embedded sources

means that you received a GPL-2.0 licensed snippet, module or library that you are now going to distribute to third parties as an embedded component

[628] For details → OSLiC, pp. 113 – 118

[629] For implementing the handover of files correctly → OSLiC, p. 127

[630] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

of a larger unit and in the form of unmodified source code files or as an unmodified source code package.

covers OSUC-07S[631]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-2.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the GPL-2.0 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the GPL-2.0 license. If it is not already part of the software package, add it.[632]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the GPL-2.0 licensed library and that it is itself licensed under the GPL-2.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.
- [**mandatory:**] Arrange the the sources of the on-top development in a way that they are covered by the GPL-2.0 licensing statements.
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.

prohibits nothing explicitely.

6.8.5 GPL-2.0-C5: Passing the unmodified library as embedded binaries

means that you received a GPL-2.0 licensed snippet, module or library that you are now going to distribute to third parties as an embedded component of a larger unit and in the form of unmodified binary files or as unmodified binary package.

631) For details → OSLiC, pp. 120

632) For implementing the handover of files correctly → OSLiC, p. 127

covers OSUC-07B[633]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-2.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the GPL-2.0 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the GPL-2.0 license. If it is not already part of the software package, add it.[634]
- [**mandatory:**] Make the *complete* source code of the program embedding the library publicly available (and, therefore, also the source code of the library itself): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the GPL-2.0 licensed library and that it is itself licensed under the GPL-2.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.
- [**mandatory:**] Arrange the the binaries of the on-top development in a way that they are covered by the GPL-2.0 licensing statements.
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Execute the to-do list of use case GPL-2.0-C4 for the source code that you publish.[635]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright

633) For details → OSLiC, pp. 121

634) For implementing the handover of files correctly → OSLiC, p. 127

635) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.

prohibits nothing explicitely.

6.8.6 GPL-2.0-C6: Passing a modified program as source code

means that you received a GPL-2.0 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

covers OSUC-04S[636]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-2.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the GPL-2.0 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the GPL-2.0 license. If it is not already part of the software package, add it.[637]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Let the copyright dialog of the program clearly say that it is a GPL-2.0 licensed program. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0. If these conditions are not already met, add the missing elements.
- [**mandatory:**] Mark all modifications of the source code the program (proapse) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the program in a way that they are covered by existing GPL-2.0 licensing statements. If you add new source code files to the program, insert a header containing

[636] For details → OSLiC, pp. 116

[637] For implementing the handover of files correctly → OSLiC, p. 127

your copyright line and a licensing statement in the form recommended by the GPL-2.0.[638]

- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.

prohibits nothing explicitely.

6.8.7 GPL-2.0-C7: Passing a modified program as binary

means that you received a GPL-2.0 licensed program, application, or server (proapse), that you modified it, and that you are now going todistribute this modified version to third parties in the form of binary files or as a binary package.

covers OSUC-04B[639]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-2.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the GPL-2.0 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the GPL-2.0 license. If it is not already part of the software package, add it.[640]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Mark all modifications of the source code the program (proapse) thoroughly within the source code and include the date of the modification.

638) For details see section 'How to Apply These Terms to Your New Programs' in the GPL-2.0 license.

639) For details → OSLiC, pp. 116

640) For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Let the copyright dialog of the program clearly say that it is a GPL-2.0 licensed program. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0. If these conditions are not already met, add the missing elements.
- [**mandatory:**] Arrange your modifications of the program in a way that they are covered by existing GPL-2.0 licensing statements. If you add new source code files to the program, insert a header containing your copyright line and a licensing statement in the form recommended by the GPL-2.0.[641]
- [**mandatory:**] Make the source code of the distributed software publicly available: Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case GPL-2.0-C6 for the source code that you publish.[642]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.

prohibits nothing explicitely.

6.8.8 GPL-2.0-C8: Passing a modified library as independent source code

means that you received a GPL-2.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

641) For details see section 'How to Apply These Terms to Your New Programs' in the GPL-2.0 license.

642) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

covers OSUC-08S[643]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-2.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the GPL-2.0 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the GPL-2.0 license. If it is not already part of the software package, add it.[644]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Mark all modifications of the source code of the library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the library in a way that they are covered by existing GPL-2.0 licensing statements. If you add new source code files to the library, insert a header containing your copyright line and a licensing statement in the form recommended by the GPL-2.0.[645]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.

prohibits nothing explicitely.

643) For details → OSLiC, pp. 122
644) For implementing the handover of files correctly → OSLiC, p. 127
645) For details see section 'How to Apply These Terms to Your New Programs' in the GPL-2.0 license.

6.8.9 GPL-2.0-C9: Passing a modified library as independent binary

means that you received a GPL-2.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers OSUC-08B[646]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-2.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the GPL-2.0 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the GPL-2.0 license. If it is not already part of the software package, add it.[647]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Make the source code of the distributed software publicly available: Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case GPL-2.0-C8 for the source code that you publish.[648]
- [**mandatory:**] Mark all modifications of the source code of the library (snimoli) thoroughly within the source code and include the date of the modification.

[646] For details → OSLiC, pp. 123

[647] For implementing the handover of files correctly → OSLiC, p. 127

[648] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

- [**mandatory:**] Arrange your modifications of the library in a way that they are covered by existing GPL-2.0 licensing statements. If you add new source code files to the library, insert a header containing your copyright line and a licensing statement in the form recommended by the GPL-2.0.[649]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.

prohibits nothing explicitely.

6.8.10 GPL-2.0-CA: Passing a modified library as embedded source code

means that you received a GPL-2.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10S[650]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-2.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the GPL-2.0 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the GPL-2.0 license. If it

[649] For details see section 'How to Apply These Terms to Your New Programs' in the GPL-2.0 license.

[650] For details → OSLiC, pp. 125

is not already part of the software package, add it.[651]

- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the GPL-2.0 licensed library and that it is itself licensed under the GPL-2.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.
- [**mandatory:**] Mark all modifications of the source code of the embedded library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the embedded library in a way that they are covered by existing GPL-2.0 licensing statements. If you add new source code files to the embedded library, insert a header containing your copyright line and a licensing statement in the form recommended by the GPL-2.0.[652]
- [**mandatory:**] Arrange the the sources of the on-top development in a way that they are covered by the GPL-2.0 licensing statements.
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.

prohibits nothing explicitely.

6.8.11 GPL-2.0-CB: Passing a modified library as embedded binary

means that you received a GPL-2.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third partiesin the form of binary files or as a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10B[653]

651) For implementing the handover of files correctly → OSLiC, p. 127
652) For details see section 'How to Apply These Terms to Your New Programs' in the GPL-2.0 license.
653) For details → OSLiC, pp. 126

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-2.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the GPL-2.0 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the GPL-2.0 license. If it is not already part of the software package, add it.[654]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Make the *complete* source code of the program embedding the library publicly available (and, therefore, also the source code of the library itself): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case GPL-2.0-CA for the source code that you publish.[655]
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the GPL-2.0 licensed library and that it is itself licensed under the GPL-2.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.
- [**mandatory:**] Mark all modifications of the source code of the embedded library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the embedded library in a way that they are covered by existing GPL-2.0 licensing statements. If you add new source code files to the embedded library, insert a

654) For implementing the handover of files correctly → OSLiC, p. 127

655) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

header containing your copyright line and a licensing statement in the form recommended by the GPL-2.0.[656]

- [**mandatory:**] Arrange the the binaries of the on-top development in a way that they are covered by the GPL-2.0 licensing statements.
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0.

prohibits nothing explicitely.

6.8.12 GPL-3.0-C1: Using the software only for yourself

means that you received GPL-3.0 licensed software, that you will use it only for yourself, and that you do not hand over to any third party in any sense.

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L and OSUC-09N[657]

requires no tasks in order to fulfill the conditions of the General Public License Version 3 with respect to this use case:

- You are allowed to use any kind of GPL software in any sense and in any context without being obliged to do anything as long as you do not give the software to third parties.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.8.13 GPL-3.0-C2: Passing the unmodified software as independent sources

means that you received GPL-3.0 licensed software that you are now going to distribute to third parties as an independent unit and in the form of unmodified source code files or as an unmodified source code package. In

656) For details see section 'How to Apply These Terms to Your New Programs' in the GPL-2.0 license.

657) For details → OSLiC, pp. 112 – 124

this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin.

covers OSUC-02S, OSUC-05S[658]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the GPL-3.0 license. If it is not already part of the software package, add it.[659]
- [**mandatory:**] Retain all existing copyright notices.
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.8.14 GPL-3.0-C3: Passing the unmodified software as independent binaries

means that you received GPL-3.0 licensed software, which you are now going to distribute to third parties as an independent unit and in the form of unmodified binary files or as an unmodified binary package. In this case it does not matter if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin.

covers OSUC-02B, OSUC-05B[660]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.

[658] For details → OSLiC, pp. 112 – 117
[659] For implementing the handover of files correctly → OSLiC, p. 127
[660] For details → OSLiC, pp. 113 – 118

- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the GPL-3.0 license. If it is not already part of the software package, add it.[661]
- [**mandatory:**] Make the source code of the distributed software publicly available (even though you did not modify it): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Execute the to-do list of use case GPL-3.0-C2 for the source code that you publish.[662]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.8.15 GPL-3.0-C4: Passing the unmodified library as embedded sources

means that you received a GPL-3.0 licensed snippet, module or library that you are now going to distribute to third parties as an embedded component of a larger unit and in the form of unmodified source code files or as an unmodified source code package.

covers OSUC-07S[663]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.

[661] For implementing the handover of files correctly → OSLiC, p. 127

[662] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

[663] For details → OSLiC, pp. 120

- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the GPL-3.0 license. If it is not already part of the software package, add it.[664]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the GPL-3.0 licensed library and that it is itself licensed under the GPL-3.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.
- [**mandatory:**] Arrange the the sources of the on-top development in a way that they are covered by the GPL-3.0 licensing statements.
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.8.16 GPL-3.0-C5: Passing the unmodified library as embedded binaries

means that you received a GPL-3.0 licensed snippet, module or library that you are now going to distribute to third parties as an embedded component of a larger unit and in the form of unmodified binary files or as unmodified binary package.

covers OSUC-07B[665]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.

[664] For implementing the handover of files correctly → OSLiC, p. 127
[665] For details → OSLiC, pp. 121

- [**mandatory:**] Give the recipient a copy of the GPL-3.0 license. If it is not already part of the software package, add it.[666]
- [**mandatory:**] Make the *complete* source code of the program embedding the library publicly available (and, therefore, also the source code of the library itself): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the GPL-3.0 licensed library and that it is itself licensed under the GPL-3.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.
- [**mandatory:**] Arrange the the binaries of the on-top development in a way that they are covered by the GPL-3.0 licensing statements.
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Execute the to-do list of use case GPL-3.0-C4 for the source code that you publish.[667]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.8.17 GPL-3.0-C6: Passing a modified program as source code

means that you received a GPL-3.0 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

666) For implementing the handover of files correctly → OSLiC, p. 127

667) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

covers OSUC-04S[668]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the GPL-3.0 license. If it is not already part of the software package, add it.[669]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Let the copyright dialog of the program clearly say that it is a GPL-3.0 licensed program. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0. If these conditions are not already met, add the missing elements.
- [**mandatory:**] Mark all modifications of the source code the program (proapse) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the program in a way that they are covered by existing GPL-3.0 licensing statements. If you add new source code files to the program, insert a header containing your copyright line and a licensing statement in the form recommended by the GPL-3.0.[670]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.

prohibits . . .

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

668) For details → OSLiC, pp. 116

669) For implementing the handover of files correctly → OSLiC, p. 127

670) For details see section 'How to Apply These Terms to Your New Programs' in the GPL-3.0 license.

6.8.18 GPL-3.0-C7: Passing a modified program as binary

means that you received a GPL-3.0 licensed program, application, or server (proapse), that you modified it, and that you are now going todistribute this modified version to third parties in the form of binary files or as a binary package.

covers OSUC-04B[671]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the GPL-3.0 license. If it is not already part of the software package, add it.[672]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Mark all modifications of the source code the program (proapse) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Let the copyright dialog of the program clearly say that it is a GPL-3.0 licensed program. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0. If these conditions are not already met, add the missing elements.
- [**mandatory:**] Arrange your modifications of the program in a way that they are covered by existing GPL-3.0 licensing statements. If you add new source code files to the program, insert a header containing your copyright line and a licensing statement in the form recommended by the GPL-3.0.[673]
- [**mandatory:**] Make the source code of the distributed software publicly available: Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.

671) For details → OSLiC, pp. 116

672) For implementing the handover of files correctly → OSLiC, p. 127

673) For details see section 'How to Apply These Terms to Your New Programs' in the GPL-3.0 license.

- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case GPL-3.0-C6 for the source code that you publish.[674]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.8.19 GPL-3.0-C8: Passing a modified library as independent source code

means that you received a GPL-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

covers OSUC-08S[675]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the GPL-3.0 license. If it is not already part of the software package, add it.[676]
- [**mandatory:**] Retain all existing copyright notices.

674) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.
675) For details → OSLiC, pp. 122
676) For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Mark all modifications of the source code of the library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the library in a way that they are covered by existing GPL-3.0 licensing statements. If you add new source code files to the library, insert a header containing your copyright line and a licensing statement in the form recommended by the GPL-3.0.[677]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.8.20 GPL-3.0-C9: Passing a modified library as independent binary

means that you received a GPL-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers OSUC-08B[678]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.

677) For details see section 'How to Apply These Terms to Your New Programs' in the GPL-3.0 license.

678) For details → OSLiC, pp. 123

- [**mandatory:**] Give the recipient a copy of the GPL-3.0 license. If it is not already part of the software package, add it.[679]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Make the source code of the distributed software publicly available: Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case GPL-3.0-C8 for the source code that you publish.[680]
- [**mandatory:**] Mark all modifications of the source code of the library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the library in a way that they are covered by existing GPL-3.0 licensing statements. If you add new source code files to the library, insert a header containing your copyright line and a licensing statement in the form recommended by the GPL-3.0.[681]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

679) For implementing the handover of files correctly → OSLiC, p. 127

680) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

681) For details see section 'How to Apply These Terms to Your New Programs' in the GPL-3.0 license.

6.8.21 GPL-3.0-CA: Passing a modified library as embedded source code

means that you received a GPL-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10S[682]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the GPL-3.0 license. If it is not already part of the software package, add it.[683]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the GPL-3.0 licensed library and that it is itself licensed under the GPL-3.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.
- [**mandatory:**] Mark all modifications of the source code of the embedded library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the embedded library in a way that they are covered by existing GPL-3.0 licensing statements. If you add new source code files to the embedded library, insert a header containing your copyright line and a licensing statement in the form recommended by the GPL-3.0.[684]
- [**mandatory:**] Arrange the the sources of the on-top development in a way that they are covered by the GPL-3.0 licensing statements.

[682] For details → OSLiC, pp. 125
[683] For implementing the handover of files correctly → OSLiC, p. 127
[684] For details see section 'How to Apply These Terms to Your New Programs' in the GPL-3.0 license.

- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.8.22 GPL-3.0-CB: Passing a modified library as embedded binary

means that you received a GPL-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third partiesin the form of binary files or as a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10B[685]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the GPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the GPL-3.0 license. If it is not already part of the software package, add it.[686]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Make the *complete* source code of the program embedding the library publicly available (and, therefore, also the source code of the library itself): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.

685) For details → OSLiC, pp. 126
686) For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case GPL-3.0-CA for the source code that you publish.[687]
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the GPL-3.0 licensed library and that it is itself licensed under the GPL-3.0, too. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.
- [**mandatory:**] Mark all modifications of the source code of the embedded library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the embedded library in a way that they are covered by existing GPL-3.0 licensing statements. If you add new source code files to the embedded library, insert a header containing your copyright line and a licensing statement in the form recommended by the GPL-3.0.[688]
- [**mandatory:**] Arrange the the binaries of the on-top development in a way that they are covered by the GPL-3.0 licensing statements.
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.8.23 Discussions and Explanations

The GPL-2.0 allows to "[...] copy and (to) distribute verbatim copies of the Program's complete source code as you receive it [...] provided that you [a]

687) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

688) For details see section 'How to Apply These Terms to Your New Programs' in the GPL-3.0 license.

conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; [b] keep intact all the notices that refer to this License and to the absence of any warranty; and [c] distribute a copy of this License along with the Program."[689] The GPL-2.0 also allows to "[...] copy and distribute [...] modifications (of the Program or any portion of it) [...] under the terms of Section 1"[690] while it allows to distribute binaries "under the terms of Sections 1 and 2".[691] But the GPL-2.0 does not require any tasks if you are using the work only for yourself. Thus, the quoted conditions of "Section 1" are mandatory for all use cases concerning the distribution of an GPL-2.0 licensed work (GPL-2.0-C2 – GPL-2.0-CB)

The GPL-3.0 uses a similar structure to establish the same requirements: In §4 it allows to "[...] convey verbatim copies of the Program's source code as you receive it [...] provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program". §5 also allows to "[...] convey [...] modifications [...] under the terms of section 4 [...]" and §6 gives permission to "[...] convey a covered work in object form under the terms of sections of 4 and 5".[692] In contrast to the GPL-2.0, the GPL-3.0 explicitly states that one "[...] may make, run and propagate covered works that (one) (does) not convey [distribute], without conditions so long as (the GPL-3.0) license otherwise remains in force."[693] Moreover, giving a package to a third party for getting a modified version back has not to be taken as a case of distribution if the modification has only been executed on behalf and only for the purpose of the purchaser and if the modified version is not distributed to any third party.[694] If one collects all these GPL-3.0 statements together, than one may conclude that the tasks which fulfill the corresponding GPL-2.0 requirements together also fit the GPL-3.0 conditions.

The GPL-2.0 allows to "[...] copy and (to) distribute the Program (or a work based on it [...]) in object code or executable form [...] provided that you accompany it with the complete corresponding machine-readable source code [...] on a medium customarily used for software interchange".[695] As a substitution for this basic condition, the GPL-2.0 allows to "accompany" the binary distribution package "[...] with a written offer, valid for at least three years, to give any third party,

689) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp §1.
690) cf. id., l.c., wp §2.
691) cf. id., l.c., wp §4.
692) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp §4, §5, §6.
693) cf. id., l.c., wp §2.
694) cf. id., ibid.
695) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp §3, §3a.

for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code [...] on a medium customarily used for software interchange".[696] The OSLiC construes the common technique to download files from the Internet as a distribution *on a medium [being today] customarily used for software interchange*. Therefore, the OSLiC requires for all open source use cases that refer to the distribution of binaries (GPL-2.0-C3, GPL-2.0-C7, GPL-2.0-C9, GPL-2.0-CB) to make the corresponding source code of the library itself accessible via an Internet repository under your own control.

The GPL-3.0 also explicitly requires to make the source code accessible in case of distributing binaries. But opposite to the GPL-2.0, the GPL-3.0 explicitly offers the option of giving "[...] access to copy the Corresponding Source from a network server at no charge" as a means to fulfill the conditions.[697] So again, the tasks which ensure to act in accordance to the GPL-2.0 license in case of distributing binaries, also fulfill the conditions of the GPL-3.0.

The weakness that in this case "third parties [which have received the binaries] are not compelled to copy the source code [...]" is a concession made by the GPL-2.0.[698] But the necessity to offer the source code via a repository controlled by yourself may generally not be circumvented: The GPL-2.0 allows to redistribute a link to an external source code repository only in case of "noncommercial distributions".[699]

Both, the GPL-2.0 and the GPL-3.0 allow you to "[...] modify your copy or copies of the Program or any portion of it [...] and (to) copy and distribute such modifications [...]" only under very similar restrictions and conditions:[700]

- First, modified files must be marked as modifications and the date of the modification.[701] These conditions must be respected by all open source use cases concerning the distribution of the modified work [GPL-2.0-C6/GPL-3.0C6 – GPL-2.0-C9/GPL-3.0-C9], because even if one primarily intends to distribute binaries, one has also to deliver the source code. The OSLiC captures this requirement in the mandatory condition to mark each modified file and the voluntary condition to update / generate a general changelog.
- Second, both versions of the GPL require that all copies of the modified software which are using an interactive interface or a method to display messages must "[...] print or display an announcement including an ap-

696) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp §3b.
697) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp §6 and §6b.
698) cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp §3, at the end.
699) cf. id., l.c., wp §3c.
700) cf. id., l.c., wp §2.
701) For GPL-2.0 see cf. id., l.c., wp. §2.
For GPL-3.0 see cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. §5.

propriate copyright notice and a notice that there is no warranty [...] and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License."[702] The OSLiC rewrites this condition in the form that the work shall let its copyright dialog clearly reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the GPL-2.0-file (or GPL-3.0-file, resp.), which has to be delivered together with the software. These conditions have to be respected if one redistributes the received and then modified programs (GPL-2.0-C6, GPL-2.0-C7, GPL-3.0-C6, GPL-3.0-C7) or if one distributes one's own programs which are using (modified) libraries as embedded components (GPL-2.0-CA, GPL-2.0-CB, GPL-3.0-CA, GPL-3.0-CB). For those open source use cases that concern the redistribution of received and modified libraries, etc., the OSLiC does not mention these requirements because libraries, plugins, or snippets normally do not have their own copyright dialogs.

- Third, the GPL requires to " [...] cause any work (being distributed or published), that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this (GPL.)"[703] This requirement does not depend of the form in which the software is distributed. The OSLiC adopts this statement in the following way:
  - For all open source use cases which concern the distribution (GPL-2.0-C2 ... GPL-2.0-CB, GPL-3.0-C2 ... GPL-3.0-CB), the OSLiC rewrites this condition as the mandatory requirement to retain all existing licensing elements.
  - For all use cases which deal with the distribution of a modified version of the software (GPL-2.0-C6 ... GPL-2.0-CB, GPL-3.0-C6 ... GPL-3.0-CB), the OSliC adds the requirement to organize the modifications in a way that they are covered by the respective GPL-2.0 or GPL-3.0 licensing statements.
  - For the use case which deal with the distribution of an embedded library (GPL-2.0-C4, GPL-2.0-C5, GPL-2.0-CA, GPL-2.0-CB, GPL-3.0-C4, GPL-3.0-C5, GPL-3.0-CA, GPL-3.0-CB) the OSLiC requires also to license the on-top development under the terms of the respective GPL-2.0 or GPL-3.0 license.
- Finally, as parts of those task lists which concern the distribution in the form of binaries, the OSLiC reminds the reader also to execute the corresponding

[702] For GPL-2.0 see cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp. §2c.
For GPL-3.0 see cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. §5d.
[703] For GPL-2.0 see cf. *Open Source Initiative*: The GPL-2.0 License (OSI), 1991, wp. §2b.
For GPL-3.0 see cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. §5c.

source code use cases because distributing the binaries without making the corresponding sources accessible is not allowed by the GPL.

And a last issue should be addressed here. It concerns the problem of granularity.

The GPL-3.0 allows "[...] to convey a covered work in object code form [...] provided that [one] also conveys the [...] Corresponding Source [...]"[704]. For understanding the scope of the sources one has to convey, one must known, what the term *Corresponding Source* means. Fortunately, the GPL-3.0 assists its readers to understand this term in the right way:

- "The 'Corresponding Source' for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities.[705]" Thus, if one took this statements seriously, one would have to "provide access to" the complete software stack of the executed AGPL program, just down to the glibc. But the GPL does not want to be to greedy. Therefore it limits the scope:
- To limit the sope, the GPL states, that the *Corresponding Source* "[...] does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work"[706]. Unfortunately, one now has to analyze, what the term *System Libraries* means, if one wants to understand this rule correctly.
- Therefore, the GPl says also, that "the 'System Libraries' of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form.[707]". And for understing this sentence adequately, one has to know, what a *Major Component* is.
- So, finally, the GPL defines as "enquoteMajor Component [...as] a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it[708]".

Based on these specifications, one can give some rule of thumbs concerning the question down to which level one has to give access to the corresponding source

704) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp §6.
705) cf. id., l.c., wp. §1.
706) cf. id., ibid.
707) cf. id., ibid.
708) cf. id., ibid.

code of an conveyed GPL binary program:

- If one conveys a GPL licensed binary program, then one has also to deliver the code of
  - the dlivered program itself
  - every modified embedded component of that program
  - every not freely accessible embedded component of that program
  - all not freely accessible tools, scripts, data which are necessary to compile the sources of the program in a freely accessible compilation / developement environment

  But it is not necessary to deliver the code of unmodified standard libraries, compilers, or tools which can freely be downloaded from their standard repository.
- If one conveys a GPL licensed script, then one has also to deliver the code of
  - every modified embedded script component included by the main script
  - every not freely accessible embedded script component included by the main script
  - all not freely accessible tools, scripts, data which are necessary to to let that main script be executed by a freely accessible interpreter
  - the interpreter itself if it is not freely accessible.

  But it is not necessary to give access to unmodified standard script libraries, interpreters, or tools which can freely be downloaded from their standard repository.

6.9 LGPL licensed software

Both versions of the GNU Lesser General Public License explicitly distinguish the distribution of the source code from that of the binaries: On the one hand, the LGPL-2.1 mainly talks about copying and distributing the source code.[709] But it also directly mentions the specific conditions for "[...] (copying) and (distributing) the Library [...] in object code or executable form [...]"[710] On the other hand, the LGPL-3.0 and the GPL-3.0—which have to be considered together because the GPL-3.0 is included into the LGPL-3.0[711]— treat the distribution of

[709] cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp §1, §2, §5, §6.
[710] cf. id., l.c., wp §4.
[711] cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp just before §0.

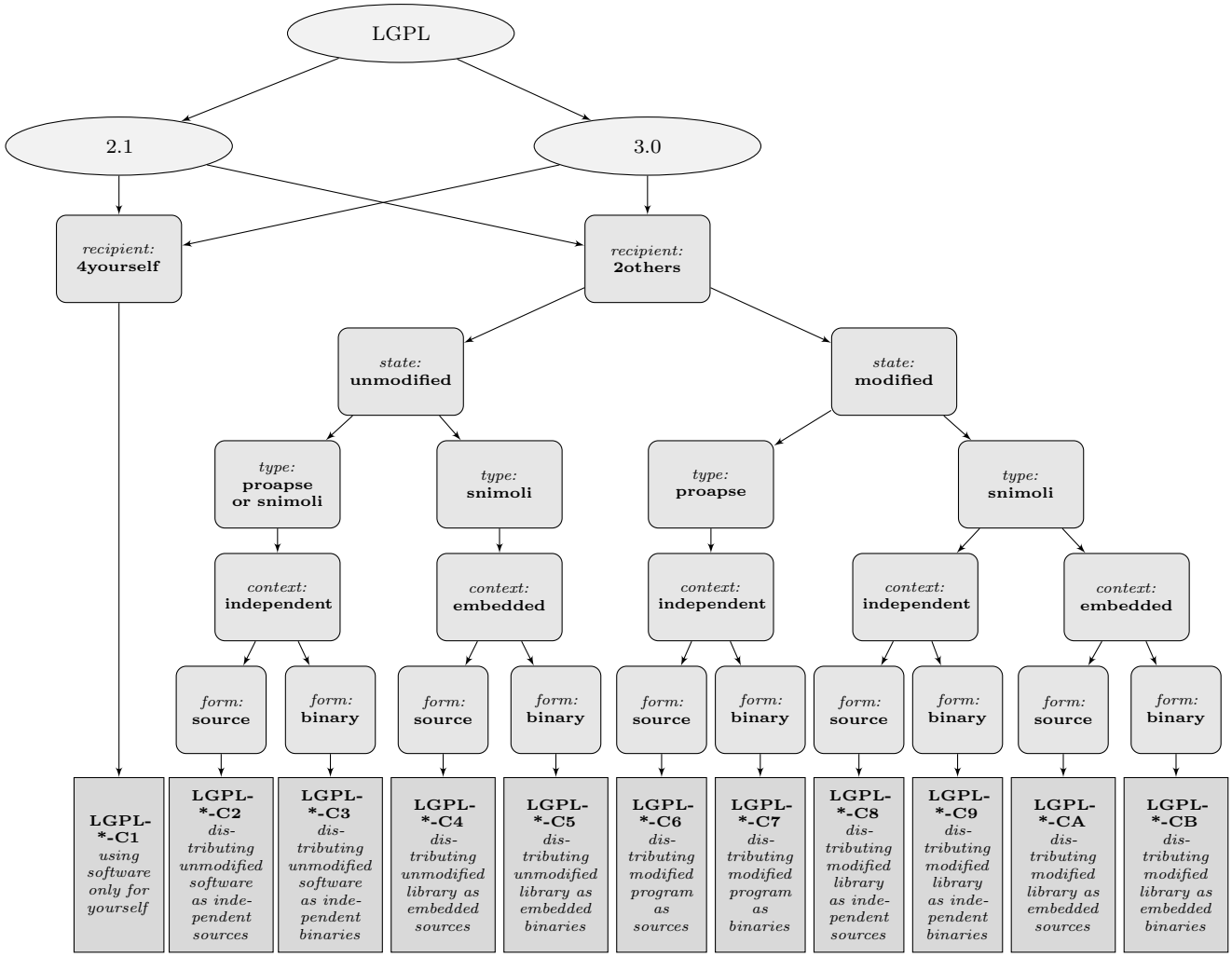
source code and the distribution of object code as different aspects of the same phenomenon[712] Additionally, LGPL-2.1 and LGPL-3.0 mainly talk about copying and distributing the software; the private use is almost complete unspecified.[713] Finally, the LGPL-2.1 and the LGPL-3.0 aim for the same results and share the same spirit by requiring nearly the same license fulfilling tasks. Therefore it seems appropriate to cover both versions in one chapter[714] and to offer the same LGPL specific open source use case structure[715] for finding the corresponding task lists:

712) The GPL-3.0 contains a specific section named "Conveying Non-Source Forms" which describes the conditions to "[...] convey a covered work in object code form [...]" (cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. §6), while the LGPL-3.0 explicitly deals with the "object code incorporating material from (the) library header files" (cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp. §3).

713) The LGPL-2.1 lists its 'restrictions' only with respect to the act of copying and distributing "copies of the library" (cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp. §1, §2, §4 et passim) while the GPL-3.0 explicitly specifies that one "[...] may make, run and propagate covered works that (one does) not convey, without conditions so long as (the) license otherwise remains in force" (cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. §2).

714) The exception concerns the distribution of a modified program, application, or server under the terms of the LGPL

715) For details of the general OSUC finder → OSLiC, pp. 104 and **??**



6.9.1 LGPL-2.1-C1: Using the software only for yourself

means that you received LGPL-2.1 licensed software, that you will use it only for yourself, and that you do not hand over to any third party in any sense.

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L and OSUC-09N[716]

requires no tasks in order to fulfill the conditions of the GNU Lesser General Public License 2.1 with respect to this use case:

- You are allowed to use any kind of LGPL-2.1 licensed software in any sense and in any context without being obliged to do anything as long as you do not give the software to third parties.

[716] For details → OSLiC, pp. 112 – 124

prohibits nothing explicitely.

6.9.2 LGPL-2.1-C2: Passing the unmodified software as independent source code

means that you received LGPL-2.1 licensed software that you are now going to distribute to third parties as an independent unit and in the form of unmodified source code files or as an unmodified source code package. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin.

covers OSUC-02S, OSUC-05S[717]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-2.1 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the LGPL-2.1 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the LGPL-2.1 license. If it is not already part of the software package, add it.[718]
- [**mandatory:**] Retain all existing copyright notices.
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-2.1.

prohibits nothing explicitely.

6.9.3 LGPL-2.1-C3: Passing the unmodified software as independent binaries

means that you received LGPL-2.1 licensed software, which you are now going to distribute to third parties as an independent unit and in the form of

[717] For details → OSLiC, pp. 112 – 117
[718] For implementing the handover of files correctly → OSLiC, p. 127

unmodified binary files or as an unmodified binary package. In this case it does not matter if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin.

covers OSUC-02B, OSUC-05B[719]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-2.1 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the LGPL-2.1 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the LGPL-2.1 license. If it is not already part of the software package, add it.[720]
- [**mandatory:**] Make the source code of the distributed software publicly available (even though you did not modify it): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**voluntary:**] Retain all existing copyright notices.
- [**mandatory:**] Execute the to-do list of use case LGPL-2.1-C2 for the source code that you publish.[721]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-2.1.

prohibits nothing explicitely.

719) For details → OSLiC, pp. 113 – 118

720) For implementing the handover of files correctly → OSLiC, p. 127

721) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

6.9.4 LGPL-2.1-C4: Passing the unmodified library as embedded source code

means that you received an LGPL-2.1 licensed snippet, module or library that you are now going to distribute to third parties as an embedded component of a larger unit and in the form of unmodified source code files or as an unmodified source code package.

covers OSUC-07S[722]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-2.1 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the LGPL-2.1 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the LGPL-2.1 license. If it is not already part of the software package, add it.[723]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-2.1.
- [**voluntary:**] Retain all existing copyright notices.

prohibits nothing explicitely.

6.9.5 LGPL-2.1-C5: Passing the unmodified library as embedded binaries

means that you received an LGPL-2.1 licensed snippet, module or library that you are now going to distribute to third parties as an embedded component of a larger unit and in the form of unmodified binary files or as unmodified binary package.

covers OSUC-07B[724]

[722] For details → OSLiC, pp. 120
[723] For implementing the handover of files correctly → OSLiC, p. 127
[724] For details → OSLiC, pp. 121

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-2.1 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the LGPL-2.1 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the LGPL-2.1 license. If it is not already part of the software package, add it.[725]
- [**mandatory:**] Make the source code of the distributed software publicly available (even though you did not modify it): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Either distribute the on-top development and the library in the form of dynamically linkable parts or distribute the statically linked application together with a written offer, valid for at least three years, to give the user all object-files of the on-top development and the library, so that he can relink the application himself.
- [**mandatory:**] Execute the to-do list of use case LGPL-2.1-C4 for the source code that you publish.[726]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-2.1.
- [**voluntary:**] Retain all existing copyright notices.

prohibits nothing explicitely.

---

[725] For implementing the handover of files correctly → OSLiC, p. 127

[726] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

6.9.6 LGPL-2.1-C6: Passing a modified program as source code

means that you received an LGPL-2.1 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

covers OSUC-04S[727]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Change all the notices in all files that refer to the LGPL-2.1, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License.

prohibits ...

- to modify the received work in a way that the resulting "modified work" is no longer a software library (but a program). **You are not allowed to distribute a modified program under the terms of LGPL-2.1.**[728]

6.9.7 LGPL-2.1-C7: Passing a modified program as binary

means that you received an LGPL-2.1 licensed program, application, or server (proapse), that you modified it, and that you are now going todistribute this modified version to third parties in the form of binary files or as a binary package.

covers OSUC-04B[729]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Change all the notices in all files that refer to the LGPL-2.1, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License.

prohibits ...

- to modify the received work in a way that the resulting "modified work" is no longer a software library (but a program). **You are not allowed to distribute a modified program under the terms of LGPL-2.1.**[730]

[727] For details → OSLiC, pp. 116

[728] The LGPL-2.1 explictly requires that "the modified work must itself be a software library" (cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp. §2a). For details → OSLiC, p. 269

[729] For details → OSLiC, pp. 116

[730] The LGPL-2.1 explictly requires that "the modified work must itself be a software library"

6.9.8 LGPL-2.1-C8: Passing a modified library as independent source code

means that you received an LGPL-2.1 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

covers OSUC-08S[731]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-2.1 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the LGPL-2.1 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the LGPL-2.1 license. If it is not already part of the software package, add it.[732]
- [**mandatory:**] Mark all modifications of the source code of the library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the library in a way that they are covered by existing LGPL-2.1 licensing statements. If you add new source code files to the library, insert a header containing your copyright line and a licensing statement in the form recommended by the LGPL-2.1.[733]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright

(cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp. §2a). For details → OSLiC, p. 269

731) For details → OSLiC, pp. 122

732) For implementing the handover of files correctly → OSLiC, p. 127

733) For details see section 'How to Apply These Terms to Your New Programs' in the LGPL-2.1 license.

notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-2.1.

- [**voluntary:**] Retain all existing copyright notices.

prohibits ...

- to modify the library in a way that it is no longer a library

6.9.9 LGPL-2.1-C9: Passing a modified library as independent binary

means that you received an LGPL-2.1 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers OSUC-08B[734]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-2.1 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the LGPL-2.1 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the LGPL-2.1 license. If it is not already part of the software package, add it.[735]
- [**mandatory:**] Make the source code of the distributed software publicly available: Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.

[734] For details → OSLiC, pp. 123
[735] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Execute the to-do list of use case LGPL-2.1-C8 for the source code that you publish.[736]
- [**mandatory:**] Mark all modifications of the source code of the library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the library in a way that they are covered by existing LGPL-2.1 licensing statements. If you add new source code files to the library, insert a header containing your copyright line and a licensing statement in the form recommended by the LGPL-2.1.[737]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-2.1.
- [**voluntary:**] Retain all existing copyright notices.

prohibits ...

- to modify the library in a way that it is no longer a library.

6.9.10 LGPL-2.1-CA: Passing a modified library as embedded source code

means that you received an LGPL-2.1 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10S[738]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-2.1 and to the absence of any warranty) are retained in your package in the form in which you have received them.

[736] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

[737] For details see section 'How to Apply These Terms to Your New Programs' in the LGPL-2.1 license.

[738] For details → OSLiC, pp. 125

- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the LGPL-2.1 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the LGPL-2.1 license. If it is not already part of the software package, add it.[739]
- [**mandatory:**] Mark all modifications of the source code of the embedded library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the embedded library in a way that they are covered by existing LGPL-2.1 licensing statements. If you add new source code files to the embedded library, insert a header containing your copyright line and a licensing statement in the form recommended by the LGPL-2.1.[740]
- [**mandatory:**] Maintain the structural independence of the library.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the LGPL-2.1 licensed library. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-2.1.
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-2.1.
- [**voluntary:**] Retain all existing copyright notices.

prohibits ...

- to modify the library in a way that it is no longer a library.

739) For implementing the handover of files correctly → OSLiC, p. 127

740) For details see section 'How to Apply These Terms to Your New Programs' in the LGPL-2.1 license.

6.9.11 LGPL-2.1-CB: Passing a modified library as embedded binary

means that you received an LGPL-2.1 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third partiesin the form of binary files or as a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10B[741]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-2.1 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice and disclaimer of warranty. If these elements are missing, add a new file containing the main copyright notice and the disclaimer of warranty in the form which is textually defined by the LGPL-2.1 license itself. (Yes, repeat the disclaimer although it is also part of the license itself and although you are required to hand the license itself over to the receiver.)
- [**mandatory:**] Give the recipient a copy of the LGPL-2.1 license. If it is not already part of the software package, add it.[742]
- [**mandatory:**] Make the source code of the embedded library publicly available: Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case LGPL-2.1-CA for the source code that you publish.[743]
- [**mandatory:**] Mark all modifications of the source code of the embedded library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the embedded library in a way that they are covered by existing LGPL-2.1 licensing statements.

[741] For details → OSLiC, pp. 126

[742] For implementing the handover of files correctly → OSLiC, p. 127

[743] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

If you add new source code files to the embedded library, insert a header containing your copyright line and a licensing statement in the form recommended by the LGPL-2.1.[744]

- [**mandatory:**] Maintain the structural independence of the library.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the LGPL-2.1 licensed library. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-2.1.
- [**mandatory:**] Either distribute the on-top development and the library in the form of dynamically linkable parts or distribute the statically linked application together with a written offer, valid for at least three years, to give the user all object-files of the on-top development and the library, so that he can relink the application himself.
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-2.1.
- [**voluntary:**] Retain all existing copyright notices.

prohibits ...

- to modify the library in a way that it is no longer a library.

6.9.12 LGPL-3.0-C1: Using the software only for yourself

means that you received LGPL-3.0 licensed software, that you will use it only for yourself, and that you do not hand over to any third party in any sense.

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L and OSUC-09N[745]

requires no tasks in order to fulfill the conditions of the GNU Lesser General Public License 3.0 with respect to this use case:

[744] For details see section 'How to Apply These Terms to Your New Programs' in the LGPL-2.1 license.

[745] For details → OSLiC, pp. 112 – 124

- You are allowed to use any kind of LGPL-3.0 licensed software in any sense and in any context without being obliged to do anything as long as you do not give the software to third parties.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.9.13 LGPL-3.0-C2: Passing the unmodified software as independent source code

means that you received LGPL-3.0 licensed software that you are now going to distribute to third parties as an independent unit and in the form of unmodified source code files or as an unmodified source code package. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin.

covers OSUC-02S, OSUC-05S[746]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the LGPL-3.0 license. If it is not already part of the software package, add it.[747]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.
- [**voluntary:**] Retain all existing copyright notices.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

[746] For details → OSLiC, pp. 112 – 117
[747] For implementing the handover of files correctly → OSLiC, p. 127

6.9.14 LGPL-3.0-C3: Passing the unmodified software as independent binaries

means that you received LGPL-3.0 licensed software, which you are now going to distribute to third parties as an independent unit and in the form of unmodified binary files or as an unmodified binary package. In this case it does not matter if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin.

covers OSUC-02B, OSUC-05B[748]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the LGPL-3.0 license. If it is not already part of the software package, add it.[749]
- [**mandatory:**] Make the source code of the distributed software publicly available (even though you did not modify it): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Execute the to-do list of use case LGPL-3.0-C2 for the source code that you publish.[750]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.

prohibits ...

[748] For details → OSLiC, pp. 113 – 118

[749] For implementing the handover of files correctly → OSLiC, p. 127

[750] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.9.15 LGPL-3.0-C4: Passing the unmodified library as embedded source code

means that you received an LGPL-3.0 licensed snippet, module or library that you are now going to distribute to third parties as an embedded component of a larger unit and in the form of unmodified source code files or as an unmodified source code package.

covers OSUC-07S[751]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the LGPL-3.0 license. If it is not already part of the software package, add it.[752]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the LGPL-3.0 licensed library. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

[751] For details → OSLiC, pp. 120
[752] For implementing the handover of files correctly → OSLiC, p. 127

6.9.16 LGPL-3.0-C5: Passing the unmodified library as embedded binaries

means that you received an LGPL-3.0 licensed snippet, module or library that you are now going to distribute to third parties as an embedded component of a larger unit and in the form of unmodified binary files or as unmodified binary package.

covers OSUC-07B[753]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the LGPL-3.0 license. If it is not already part of the software package, add it.[754]
- [**mandatory:**] Make the source code of the distributed software publicly available (even though you did not modify it): Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the LGPL-3.0 licensed library. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.
- [**mandatory:**] Either distribute the on-top development and the library in the form of dynamically linkable parts or distribute the statically linked application together with a written offer, valid for at least three years, to give the user all object-files of the on-top development and the library, so that he can relink the application himself.
- [**mandatory:**] Execute the to-do list of use case LGPL-3.0-C4 for the source code that you publish.[755]

[753] For details → OSLiC, pp. 121

[754] For implementing the handover of files correctly → OSLiC, p. 127

[755] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.
- [**voluntary:**] Retain all existing copyright notices.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.9.17 LGPL-3.0-C6: Passing a modified program as source code

means that you received an LGPL-3.0 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

covers OSUC-04S[756]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the LGPL-3.0 license. If it is not already part of the software package, add it.[757]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Let the copyright dialog of the program clearly say that it is a LGPL-3.0 licensed program. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0. If these conditions are not already met, add the missing elements.
- [**mandatory:**] Mark all modifications of the source code the program (proapse) thoroughly within the source code and include the date of the modification.

[756] For details → OSLiC, pp. 116
[757] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Arrange your modifications of the program in a way that they are covered by existing LGPL-3.0 licensing statements. If you add new source code files to the program, insert a header containing your copyright line and a licensing statement in the form recommended by the LGPL-3.0.[758]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.9.18 LGPL-3.0-C7: Passing a modified program as binary

means that you received an LGPL-3.0 licensed program, application, or server (proapse), that you modified it, and that you are now going todistribute this modified version to third parties in the form of binary files or as a binary package.

covers OSUC-04B[759]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the LGPL-3.0 license. If it is not already part of the software package, add it.[760]
- [**mandatory:**] Retain all existing copyright notices.

[758] For details see section 'How to Apply These Terms to Your New Programs' in the LGPL-3.0 license.
[759] For details → OSLiC, pp. 116
[760] For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Mark all modifications of the source code the program (proapse) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Let the copyright dialog of the program clearly say that it is a LGPL-3.0 licensed program. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0. If these conditions are not already met, add the missing elements.
- [**mandatory:**] Arrange your modifications of the program in a way that they are covered by existing LGPL-3.0 licensing statements. If you add new source code files to the program, insert a header containing your copyright line and a licensing statement in the form recommended by the LGPL-3.0.[761]
- [**mandatory:**] Make the source code of the distributed software publicly available: Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case LGPL-3.0-C4 for the source code that you publish.[762]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

761) For details see section 'How to Apply These Terms to Your New Programs' in the LGPL-3.0 license.

762) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

6.9.19 LGPL-3.0-C8: Passing a modified library as independent source code

means that you received an LGPL-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

covers OSUC-08S[763]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the LGPL-3.0 license. If it is not already part of the software package, add it.[764]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Mark all modifications of the source code of the library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the library in a way that they are covered by existing LGPL-3.0 licensing statements. If you add new source code files to the library, insert a header containing your copyright line and a licensing statement in the form recommended by the LGPL-3.0.[765]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.

prohibits ...

[763] For details → OSLiC, pp. 122

[764] For implementing the handover of files correctly → OSLiC, p. 127

[765] For details see section 'How to Apply These Terms to Your New Programs' in the LGPL-3.0 license.

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.9.20 LGPL-3.0-C9: Passing a modified library as independent binary

means that you received an LGPL-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers OSUC-08B[766]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the LGPL-3.0 license. If it is not already part of the software package, add it.[767]
- [**mandatory:**] Retain all existing copyright notices.
- [**mandatory:**] Make the source code of the distributed software publicly available: Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case LGPL-3.0-C8 for the source code that you publish.[768]
- [**mandatory:**] Mark all modifications of the source code of the library (snimoli) thoroughly within the source code and include the date of the modification.

[766] For details → OSLiC, pp. 123

[767] For implementing the handover of files correctly → OSLiC, p. 127

[768] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

- [**mandatory:**] Arrange your modifications of the library in a way that they are covered by existing LGPL-3.0 licensing statements. If you add new source code files to the library, insert a header containing your copyright line and a licensing statement in the form recommended by the LGPL-3.0.[769]
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.9.21 LGPL-3.0-CA: Passing a modified library as embedded source code

means that you received an LGPL-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10S[770]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed source code package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the LGPL-3.0 license. If it is not already part of the software package, add it.[771]

769) For details see section 'How to Apply These Terms to Your New Programs' in the LGPL-3.0 license.
770) For details → OSLiC, pp. 125
771) For implementing the handover of files correctly → OSLiC, p. 127

- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the LGPL-3.0 licensed library. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.
- [**mandatory:**] Mark all modifications of the source code of the embedded library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the embedded library in a way that they are covered by existing LGPL-3.0 licensing statements. If you add new source code files to the embedded library, insert a header containing your copyright line and a licensing statement in the form recommended by the LGPL-3.0.[772]
- [**mandatory:**] Maintain the structural independence of the library.
- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.
- [**voluntary:**] Retain all existing copyright notices.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.9.22 LGPL-3.0-CB: Passing a modified library as embedded binary

means that you received an LGPL-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third partiesin the form of binary files or as a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10B[773]

requires the following tasks in order to fulfill the license conditions:

[772] For details see section 'How to Apply These Terms to Your New Programs' in the LGPL-3.0 license.

[773] For details → OSLiC, pp. 126

- [**mandatory:**] Ensure that the licensing elements (especially all notices that refer to the LGPL-3.0 and to the absence of any warranty) are retained in your package in the form in which you have received them.
- [**mandatory:**] Ensure that the distributed binary package contains a conspicuous, easy to find copyright notice. If this element is missing, add a new file containing the main copyright notice.
- [**mandatory:**] Give the recipient a copy of the LGPL-3.0 license. If it is not already part of the software package, add it.[774]
- [**mandatory:**] Make the source code of the embedded library publicly available: Push the source code package into a repository under your control and make it downloadable via the Internet. Ensure, that this repository is online for at least 3 years after you ceased distributing the software package.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case LGPL-3.0-CA for the source code that you publish.[775]
- [**mandatory:**] Let the copyright dialog of the on-top development clearly say that it uses the LGPL-3.0 licensed library. Let it reproduce the content of the existing copyright notices, the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.
- [**mandatory:**] Mark all modifications of the source code of the embedded library (snimoli) thoroughly within the source code and include the date of the modification.
- [**mandatory:**] Arrange your modifications of the embedded library in a way that they are covered by existing LGPL-3.0 licensing statements. If you add new source code files to the embedded library, insert a header containing your copyright line and a licensing statement in the form recommended by the LGPL-3.0.[776]
- [**mandatory:**] Maintain the structural independence of the library.
- [**mandatory:**] Either distribute the on-top development and the library in the form of dynamically linkable parts or distribute the statically linked application together with a written offer, valid for at least three

[774] For implementing the handover of files correctly → OSLiC, p. 127

[775] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

[776] For details see section 'How to Apply These Terms to Your New Programs' in the LGPL-3.0 license.

years, to give the user all object-files of the on-top development and the library, so that he can relink the application himself.

- [**voluntary:**] Create a *modification text file,* if such a file does not yet exist. *Add* a description of your modifications on a functional level to the *modification text file.*
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing copyright notices, a hint to the software name, a link to its homepage, the respective disclaimer of warranty, and a link to the LGPL-3.0.
- [**voluntary:**] Retain all existing copyright notices.

prohibits ...

- to institute a patent litigation against anyone alleging that the software constitutes patent infringement.

6.9.23 Discussions and Explanations

- The LGPL-2.1 allows to "[...] copy and (to) distribute verbatim copies of the Library's complete source code as you receive it [...] provided that you [a] conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; [b] keep intact all the notices that refer to this License and to the absence of any warranty; and [c] distribute a copy of this License along with the Library."[777] Additionally, the LGPL-2.1 allows the distribution of the modified source code "under the terms of Section 1"[778] and the distribution of binaries "under the terms of Sections 1 and 2".[779] But the LGPL does not require any tasks if you are using the work only for yourself. Thus, the quoted conditions of "Section 1" are mandatory for all use cases concerning the distribution of an LGPL licensed work (LGPL-2.1-C2 – LGPL-2.1-CB). [780]
- Although the LGPL-2.1 does not explicitly require to retain the copyright notices in the form you have received them, it is nevertheless a very good idea not to modify these elements (LGPL-2.1-C2 - LGPL-2.1-CB). The LGPL-3.0, on the other hand, inherits the clauses that require all notices to be kept intact from the GPL-3.0 (LGPL-3.0-C2 – LGPL-3.0-CB).[781]

777) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp §1.
778) cf. id., l.c., wp §2.
779) cf. id., l.c., wp §4.
780) The GPL-3.0, which is included into the LGPL-3.0, uses a similar structure to establish the same requirements (→ OSLiC, p. 237). Based on this fact, one may conclude that the tasks which fulfill the corresponding LGPL-2.1 requirements together also fit the GPL-3.0 conditions and hence those of the LGPL-3.0.
781) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp §4.

- The LGPL-2.1 allows to "[...] copy and (to) distribute the Library (or a portion or derivative of it [...]) in object code or executable form [...] provided that you accompany it with the complete corresponding machine-readable source code [...] on a medium customarily used for software interchange." And the license further states that, if one makes the object code accessible without distributing it directly, then the same 'download' method for the source code fulfills this condition.[782] So, no doubt: Taken literally, the LGPL requires you to distribute the source code and the object code together and by the same method: either both on (for example) DVD or both offered for download; but not the one on DVD and the other by a download from a repository. But the first specification also says, that the "complete corresponding machine readable source code" has to be distributed "on a medium customarily used for software interchange."[783] The OSLiC considers the possibility to download files from the Internet as a distribution *on a medium [today] customarily used for software interchange.* Therefore, the OSLiC requires for all open source use cases that refer to the distribution of binaries (LGPL-2.1-C3, LGPL-2.1-C5, LGPL-2.1-C7, LGPL-2.1-C9, and LGPL-2.1-CA) to make the source code of the corresponding library accessible via an Internet repository.

  In contrast to the LGPL-2.1, the GPL-3.0, which is included in the LGPL-3.0, explictily offers the option to distribute the sources via an Internet server (→ OSLiC, p. 238). So, one may again conclude that the tasks that fulfill the corresponding LGPL-2.1 requirements together also fit the GPL-3.0 and the LGPL-3.0 conditions.

- The LGPL allows to "[...] modify your copy or copies of the Library or any portion of it [...] and (to) copy and distribute such modifications [...]" only under some restrictions and condtions:[784]

  - First, modified files must be marked as modifications and this must include the date of the modification.[785] This condition must be respected by all open source use cases concerning the distribution of the modified work [LGPL-*-C6 - LGPL-*-CB], because even if one primarily intends to distribute binaries, one has also to deliver the source code. The OSLiC 'replaces' this requirement by the mandatory condition to mark each modified file and by the voluntary condition to update or create a general changelog file.

782) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp §4.
783) cf. id., ibid.
784) cf. id., l.c., wp §2.
785) For LGPL-2.1 see cf. id., l.c., wp. §2.
For GPL-3.0, which is included in the LGPL-3.0, see cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp. §5.

  - Second, the license requires that the modified version does not depend on external data structures without "[...] (making) a good faith effort to ensure that, in the event an application does not supply such (a) function or table, the facility still operates, and performs whatever part of its purpose remains meaningful."[786] The OSLiC rewrites this condition as the obligation to maintain the structural independence of the library in case of using the modified library as embedded component [LGPL-*-CA - LGPL-*-CB].
  - Third, the LGPL-2.1 definitely requires, that "the modified work must itself be a software library."[787] This conditions can directly be incorprated as an interdiction into all use cases which refer to the modification of a library [LGPL-2.1-C8 - LGPL-2.1-CB]. But it is difficult to respect this condition if one wants to modify a program which one has received under the terms of the LGPL-2.1. In principal, one can write an application and license it under the LGPL-2.1. But, as a consequence, that impedes the modification of this work because the result must be a library.

    The LGPL-3.0 does not contain any such requirement. Hence, the OSLiC allows the distribution of modified programs (LGPL-*-C6, LGPL-*-C7) only if they are licensed under the terms of LGPL-3.0. For programs licensed under LGPL-2.1, the only option is to relicense the software under the terms of the regular GPL-2.0 (or, at your discretion, GPL-3.0). This is explicitely allowed by the LGPL-2.1: "You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library."[788]

- Additionally, the LGPL-2.1 allows the licensee to distribute a program[789] developed on-top of the library (what the LGPL-2.1 calls a "work that uses the libary"[790]) "as an exception to the Sections above" in "combination" with the library "under terms of your choice,"[791], provided that the licensee fulfills additional conditions:

  First, it must clearly be stated that the on-top development depends on the (modified) library. Second, the LGPL must be added into the distributed package.[792] In the LGPL-3.0, this condition is similarily integrated: On the one hand, the "combined work" is defined as "a work produced by

[786] For LGPL-2.1 see cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp. §2d. For LGPL-3.0 see cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp. §2a.
[787] cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp §2.
[788] cf. id., l.c., wp §3.
[789] or another library
[790] cf. id., l.c., wp §5, §6.
[791] cf. id., l.c., wp §6.
[792] cf. id., ibid.

combining or linking an Application with the Library".[793] On the other hand, the LGPL-3.0 states that one "[...] may convey a Combined Work under terms of (his own) choice" provided that one [a] clearly says that the on-top development uses the LGPL licensed library, [b] distributes the LGPL-3.0 and the GPL-3.0 license as part of the package, [c] includes all these (licensing) information in an existing copyright dialog, if any, [d] requires an appropriate shared library mechanism, and [e] offers the respective installion information.[794] These requirements can directly be inserted as conditions into the respective use cases for both LGPL versions (LGPL-*-CA, LGPL-*-CB).

- The most difficult requirements of the LGPL-2.1 concern the distribution in the form of binaries. In a very strict reading, the LGPL does not require to link the on-top development and the libary only dynamically. At first, the LGPL mentions, that the "[..] work (that uses the Library), in isolation, is not a derivative work of the Library [...]". But if it is linked to the library the resulting executable program becomes "a derivative of the Library" and that it is therefore "[...] covered by this License (LGPL-2.1)". But the LGPL-2.1 directly continues this statement with the hint, that "Section 6 states terms for distribution of such executables."[795] Finally, section 6 directly starts with the statement: "As an exception to the Sections above, you may also combine or link a 'work that uses the Library' with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice".[796]

  This is important to know, because until this section 6 one can not directly read or indirectly infer that the LGPL-2.1 distinguished the act of dynamically linking a program and a library from that of statically linking these parts. The LGPL only wants to ensure that the binaries of the library itself can be replaced by a newer version. And that is required by section 6.[797] From a practical point of view, this can only be guaranteed, if the binaries of the on-top development and the library are linked using a "suitable shared library mechanism"[798] or if one also gets all compiled, but not linked object-files of the on-top development and the library, either directly, or via using a "a written offer, valid for at least three years, to give the same user the (respective) materials".[799] In the first case, the user can replace the received version of the library and can let the application be relinked

793) cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp §0.
794) cf. id., l.c., wp §4.
795) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp §5.
796) cf. id., l.c., wp §6.
797) cf. id., ibid.
798) cf. id., ibid.
799) cf. id., ibid.

automatically. In the second case, he has to do it manually. It is important to know that both these ways exist if one wants or must distribute statically linked works. The LGPL-2.1 does not forbid to distribute statically linked applications. But it requires to enable the receiver to relink the work.

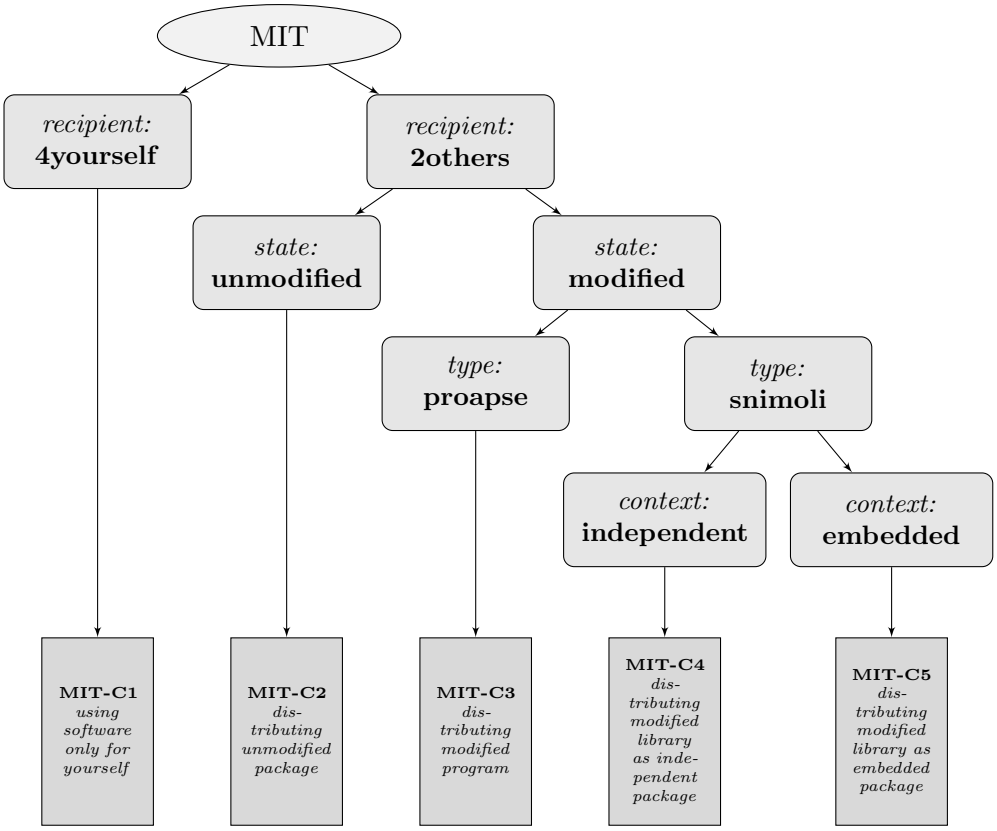
The LGPL-3.0 has reduced these complex conditions in a special way: First, it does not use the words 'statically linked' or 'dynamically' linked at all. Second it defines the combined work 'only' as the result of "combining or linking an Application with the Library".[800] But then it requires for the distribution of the combined works that one has either to "convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work [...]" or that one must presuppose that the receiver uses "[...] suitable shared library mechanism for linking with the Library [...] that [...] operate properly with a modified version of the Library [...]"[801] Finally, the LGPL-3.0 adds that in the first case the these materials which enables the relinking must be distributed "[...] in the manner specified by section 6 of the GNU GPL[-3.0] for conveying Corresponding Source."[802] And this section 6 of the GPL-3.0 allows the well known method to "convey the object code [...] accompanied by a written offer [...] to give anyone [...] access to copy the Corresponding Source from a network server at no charge".[803]

Therefore, the OSLiC can condense these conditions into the requirement, either to distribute dynamically linkable parts, or to distribute statically linked applications "(accompanied) [...] with a written offer, valid for at least three years, to give the same user the [complete] materials,"[804] so that he can relink the application. It is clear, that this condition only applies to the use cases LGPL-*-C5 and LGPL-*-CB.

6.10 MIT licensed software

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

800) cf. *Open Source Initiative*: The LGPL-3.0 License (OSI), 2007, wp §0.
801) cf. id., l.c., wp §4.
802) cf. id., ibid.
803) cf. *Open Source Initiative*: The GPL-3.0 License (OSI), 2007, wp §6.
804) cf. *Open Source Initiative*: The LGPL-2.1 License (OSI), 1999, wp §6.



6.10.1 MIT-C1: Using the software only for yourself

means that you received MIT licensed software, that you will use it only for yourself and that you do not hand it over to any 3rd party in any sense.

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L, and OSUC-09N[805]

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

- You are allowed to use any kind of MIT licensed software in any sense and in any context without any obligations if you do not give the software to third parties and if you do not modify the existing copyright notices and the existing permission notice.

prohibits nothing explicitely.

[805] For details → OSLiC, pp. 112 - 124

6.10.2 MIT-C2: Passing the unmodified software

means that you received MIT licensed software which you are now going to distribute to third parties in the form of unmodified binaries or as unmodifed source code files. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent package.

covers OSUC-02S, OSUC-02B, OSUC-05S, OSUC-05B, OSUC-07S, OSUC-07B[806]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially 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 explicitely.

6.10.3 MIT-C3: Passing a modified program

means that you received an MIT licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form binaries or as source code files.

covers OSUC-04S, OSUC-04B[807]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially 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 source code, 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 can augment an existing copyright notice presented by the program with information about your own work or modifications.

[806] For details → OSLiC, pp. 112 - 121
[807] For details → OSLiC, pp. 116

- [**voluntary:**] It is a good practice of the open source community to let the copyright notice that is shown by the running program also state that the program uses a component licensed under the MIT license. And it is a good tradition to insert links to the homepage or download page of this component.

prohibits nothing explicitely.

6.10.4 MIT-C4: Passing a modified library independently

means that you received an MIT licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the the form of binaries or as source code files, but without embedding it into another larger software unit.

covers OSUC-08S, OSUC-08B[808]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially 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 source code, 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.

prohibits nothing explicitely.

6.10.5 MIT-C5: Passing a modified library as embedded component

means that you received an MIT licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binaries or as source code files together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component, regardless whether you distribute it in the form of binaries or as source code files.

covers OSUC-10S, OSUC-10B[809]

requires the following tasks in order to fulfill the license conditions:

[808] For details → OSLiC, pp. 122
[809] For details → OSLiC, pp. 125

- [**mandatory:**] Ensure that the licensing elements (especially 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 source code, regardless whether you want to distribute the code or not.
- [**voluntary:**] It is a good practice of the open source community to let the copyright notice that is shown by the running program also state that the program uses a component licensed under the MIT license. And it is a good tradition to insert links to the homepage or download page of this component.
- [**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:**] Arrange your distribution so that the original licensing elements (especially 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 separate directories, which contain also all licensing elements.

prohibits nothing explicitely.

6.10.6 Discussions and Explanations

The MIT-License is known as one of the most permissive licenses. It is a very short license containing (0) a copyright notice, (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 disclaimer.[810] 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 readings:

- 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.

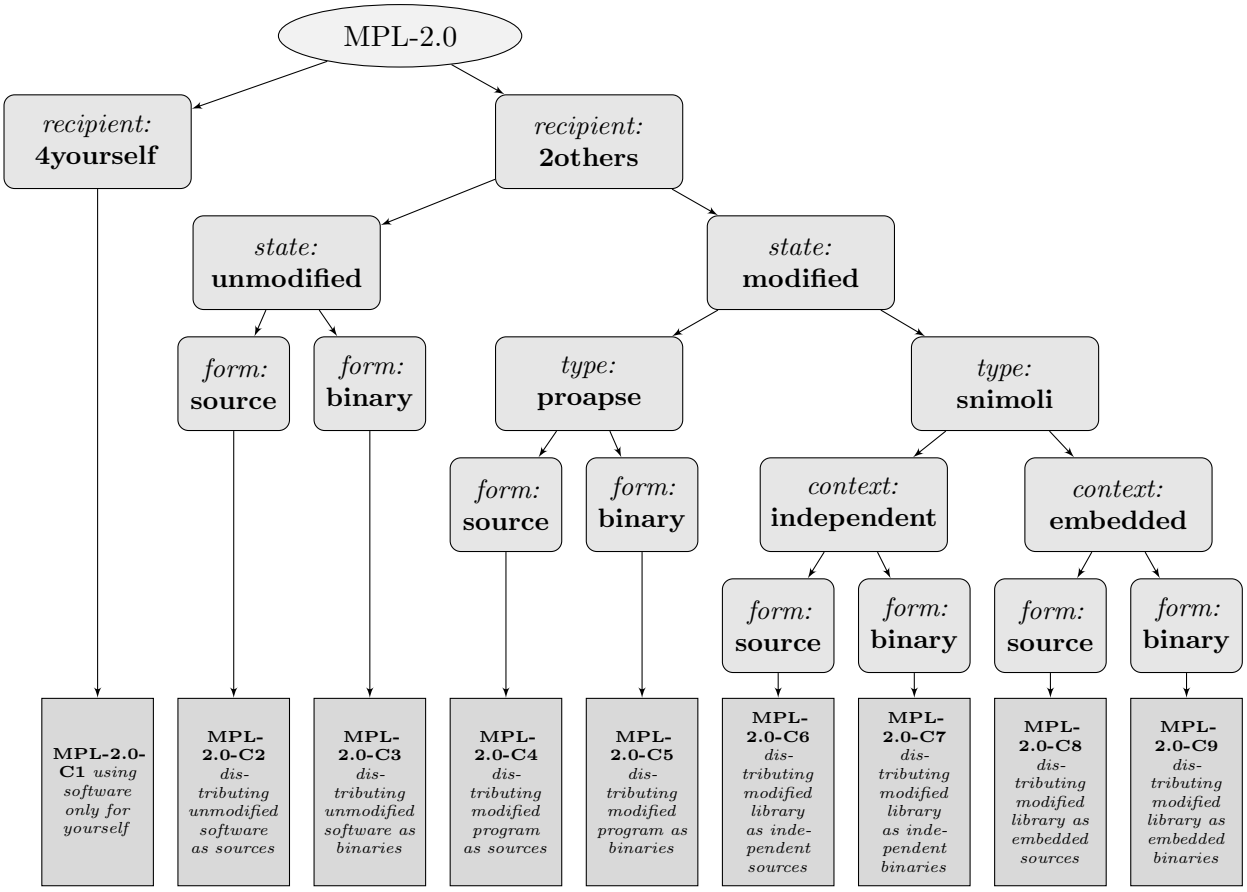
[810] cf. *Open Source Initiative*: The MIT License, 2012, wp.

- Nevertheless, we added the hint not to modify these licensing elements in the context of the use case *used by yourself*. This is implied by the MIT license itself. It requires explicitly that "the above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software"[811]–thus also into those copies you make for your own purposes on your own machines.
- If you modify the MIT licensed application, regardless for which purpose, 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 notices must survive.
- We request that you also keep the MIT disclaimer. This is not explicitely required by the license. The permission notices, which is required to be preserved, most likely refers to the text *between* the copyright notice and the disclaimer and, hence, does not include the latter. But another possible, although less likely interpretation is that the whole text of the license is what permission notice refers to.

6.11 MPL-2.0 licensed software

The Mozilla Public License clearly distinguishes the distribution of source code from the distribution of binaries: First, it allows the "Distribution of Source Form".[812] Then, it specifies the conditions for a "Distribution of Executable Form".[813] Additionally, the MPL-2.0 contrasts the "distribution of Covered Software" with the "distribution of a Larger Work".[814] So, taken as whole, the MPL-2.0 mainly focusses on the distribution of software. Thus, for finding the relevant executable task lists, the following MPL-2.0 specific open source use case structure[815] can be used:

811) cf. *Open Source Initiative*: The MIT License, 2012, wp -.
812) cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp §3.1.
813) cf. id., l.c., wp §3.2.
814) cf. id., l.c., wp §3.3.
815) For details of the general OSUC finder → OSLiC, pp. 104 and **??**



6.11.1 MPL-2.0-C1: Using the software only for yourself

means that you received MPL-2.0 licensed software, that you will use it only for yourself, and that you do not hand it over to any third party in any sense.

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L, and OSUC-09N[816]

requires no tasks in order to fulfill the conditions of the Mozilla Public License 2.0 with respect to this use case:

- You are allowed to use any kind of MPL-2.0 software in any sense and in any context without being obliged to do anything as long as you do not give the software to third parties.

prohibits ...

- to remove or to alter any license elements (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the software package you have received.

816) For details → OSLiC, pp. 112 – 124

- to promote any of your services based on the this software by trademarks, service marks, or logos linked to this MPL-2.0 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.11.2 MPL-2.0-C2: Passing the unmodified software as source code

means that you received MPL-2.0 licensed software which you are now going to distribute to third parties in the form of unmodified source code files or as unmodified source code package. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or as an embedded unit.

covers OSUC-02S, OSUC-05S, OSUC-07S[817]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all copyright notices, patent notices, disclaimers of warranty, or limitations of liability) are retained in your package in exactly the form that you have received.
- [**mandatory:**] Give the recipient a copy of the MPL-2.0 license. If it is not already part of the software package, add it. If the licensing statement in the licensing file of the package does still not clearly state that the package is licensed under the MPL-2.0, additionally insert your own correct MPL-2.0 licensing file containing the sentence: *This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.*[818]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing *copyright notice text files,* the name of the software, a link to its homepage, and a link to the MPL-2.0 license.

prohibits ...

- to remove or to alter any license elements (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the software package you have received.
- to promote any of your services based on the this software by trademarks, service marks, or logos linked to this MPL-2.0 software, except

[817] For details → OSLiC, pp. 112 – 120
[818] For implementing the handover of files correctly → OSLiC, p. 127

as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.11.3 MPL-2.0-C3: Passing the unmodified software as binaries

means that you received MPL-2.0 licensed software which you are now going to distribute to third parties in the form of unmodified binary files or as unmodified binary package. In this case it does not matter if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit.

covers OSUC-02B, OSUC-05B, OSUC-07B[819]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all copyright notices, patent notices, disclaimers of warranty, or limitations of liability) are retained in your package in exactly the form that you have received. If you compile the binary from the sources, ensure that all these licensing elements are also incorporated into the package.
- [**mandatory:**] Make the source code of the distributed software accessible via a repository under your own control: Push the source code package into the repository and make it downloadable via the Internet. Do no charge any fees from the user for downloading the source. Ensure, that this repository is online for a reasonable period of time after you ceased distributing the software.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case MPL-2.0-C2 for the source code that you publish.[820]
- [**voluntary:**] Give the recipient a copy of the MPL-2.0 license. If it is not already part of the software package, add it. If the licensing statement in the licensing file of the package does still not clearly state that the package is licensed under the MPL-2.0, additionally insert your own correct MPL-2.0 licensing file containing the sentence: *This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.*[821]

819) For details → OSLiC, pp. 113 – 121

820) Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

821) For implementing the handover of files correctly → OSLiC, p. 127

- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing *copyright notice text files,* the name of the software, a link to its homepage, and a link to the MPL-2.0 license.

prohibits ...

- to remove or to alter any license elements (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the software package you have received.
- to promote any of your services based on the this software by trademarks, service marks, or logos linked to this MPL-2.0 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.11.4 MPL-2.0-C4: Passing a modified program as source code

means that you received an MPL-2.0 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

covers OSUC-04S[822]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all copyright notices, patent notices, disclaimers of warranty, or limitations of liability) are retained in your package in exactly the form that you have received.
- [**mandatory:**] Give the recipient a copy of the MPL-2.0 license. If it is not already part of the software package, add it. If the licensing statement in the licensing file of the package does still not clearly state that the package is licensed under the MPL-2.0, additionally insert your own correct MPL-2.0 licensing file containing the sentence: *This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.*[823]
- [**mandatory:**] Organize your modifications in such a way that they are covered by the existing MPL-2.0 licensing statements. If you add new source code files, insert a header containing your copyright line and an MPL-2.0 adequate licensing the statement.

[822] For details → OSLiC, pp. 116
[823] For implementing the handover of files correctly → OSLiC, p. 127

- [**voluntary:**] Create a *modification text file*, if such a notice file still does not exist. *Add* a general description of your modifications to the *modification text file*. Incorporate the file into your distribution package.
- [**voluntary:**] Mark all modifications of the source code thoroughly, preferably in the modified source itself.
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing *copyright notice text files,* the name of the software, a link to its homepage, and a link to the MPL-2.0 license.

prohibits ...

- to remove or to alter any license elements (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the software package you have received.
- to promote any of your services based on the this software by trademarks, service marks, or logos linked to this MPL-2.0 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.11.5 MPL-2.0-C5: Passing a modified program as binary

means that you received an MPL-2.0 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package.

covers OSUC-04B[824]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all copyright notices, patent notices, disclaimers of warranty, or limitations of liability) are retained in your package in exactly the form that you have received. If you compile the binary from the sources, ensure that all these licensing elements are also incorporated into the package.
- [**mandatory:**] Make the source code of the distributed software accessible via a repository under your own control: Push the source code package into the repository and make it downloadable via the Internet. Do no charge any fees from the user for downloading the

[824] For details → OSLiC, pp. 116

source. Ensure, that this repository is online for a reasonable period of time after you ceased distributing the software.

- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case MPL-2.0-C4 for the source code that you publish.[825]
- [**mandatory:**] Organize your modifications in such a way that they are covered by the existing MPL-2.0 licensing statements.
- [**voluntary:**] Create a *modification text file*, if such a notice file still does not exist. *Add* a general description of your modifications to the *modification text file*. Incorporate the file into your distribution package.
- [**voluntary:**] Give the recipient a copy of the MPL-2.0 license. If it is not already part of the software package, add it. If the licensing statement in the licensing file of the package does still not clearly state that the package is licensed under the MPL-2.0, additionally insert your own correct MPL-2.0 licensing file containing the sentence: *This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.*[826]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing *copyright notice text files,* the name of the software, a link to its homepage, and a link to the MPL-2.0 license.

prohibits ...

- to remove or to alter any license elements (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the software package you have received.
- to promote any of your services based on the this software by trademarks, service marks, or logos linked to this MPL-2.0 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

[825] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

[826] For implementing the handover of files correctly → OSLiC, p. 127

6.11.6 MPL-2.0-C6: Passing a modified library as independent source code

means that you received an MPL-2.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

covers OSUC-08S[827]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all copyright notices, patent notices, disclaimers of warranty, or limitations of liability) are retained in your package in exactly the form that you have received.
- [**mandatory:**] Give the recipient a copy of the MPL-2.0 license. If it is not already part of the software package, add it. If the licensing statement in the licensing file of the package does still not clearly state that the package is licensed under the MPL-2.0, additionally insert your own correct MPL-2.0 licensing file containing the sentence: *This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.*[828]
- [**mandatory:**] Organize your modifications in such a way that they are covered by the existing MPL-2.0 licensing statements. If you add new source code files, insert a header containing your copyright line and an MPL-2.0 adequate licensing the statement.
- [**voluntary:**] Create a *modification text file*, if such a notice file still does not exist. *Add* a general description of your modifications to the *modification text file*. Incorporate the file into your distribution package.
- [**voluntary:**] Mark all modifications of the source code thoroughly, preferably in the modified source itself.
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing *copyright notice text files,* the name of the software, a link to its homepage, and a link to the MPL-2.0 license.

prohibits ...

[827] For details → OSLiC, pp. 122
[828] For implementing the handover of files correctly → OSLiC, p. 127

- to remove or to alter any license elements (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the software package you have received.
- to promote any of your services based on the this software by trademarks, service marks, or logos linked to this MPL-2.0 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.11.7 MPL-2.0-C7: Passing a modified library as independent binary

means that you received an MPL-2.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers OSUC-08B[829]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all copyright notices, patent notices, disclaimers of warranty, or limitations of liability) are retained in your package in exactly the form that you have received. If you compile the binary from the sources, ensure that all these licensing elements are also incorporated into the package.
- [**mandatory:**] Make the source code of the distributed software accessible via a repository under your own control: Push the source code package into the repository and make it downloadable via the Internet. Do no charge any fees from the user for downloading the source. Ensure, that this repository is online for a reasonable period of time after you ceased distributing the software.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case MPL-2.0-C6 for the source code that you publish.[830]
- [**mandatory:**] Organize your modifications in such a way that they are covered by the existing MPL-2.0 licensing statements.
- [**voluntary:**] Create a *modification text file*, if such a notice file still does not exist. *Add* a general description of your modifications to

[829] For details → OSLiC, pp. 123

[830] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

the *modification text file*. Incorporate the file into your distribution package.

- [**voluntary:**] Give the recipient a copy of the MPL-2.0 license. If it is not already part of the software package, add it. If the licensing statement in the licensing file of the package does still not clearly state that the package is licensed under the MPL-2.0, additionally insert your own correct MPL-2.0 licensing file containing the sentence: *This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.*[831]
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing *copyright notice text files,* the name of the software, a link to its homepage, and a link to the MPL-2.0 license.

prohibits ...

- to remove or to alter any license elements (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the software package you have received.
- to promote any of your services based on the this software by trademarks, service marks, or logos linked to this MPL-2.0 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.11.8 MPL-2.0-C8: Passing a modified library as embedded source code

means that you received an MPL-2.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10S[832]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all copyright notices, patent notices, disclaimers of warranty, or limitations of liability) are retained in your package in exactly the form that you have received.

[831] For implementing the handover of files correctly → OSLiC, p. 127
[832] For details → OSLiC, pp. 125

- [**mandatory:**] Give the recipient a copy of the MPL-2.0 license. If it is not already part of the software package, add it. If the licensing statement in the licensing file of the package does still not clearly state that the package is licensed under the MPL-2.0, additionally insert your own correct MPL-2.0 licensing file containing the sentence: *This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.*[833]
- [**mandatory:**] Organize your modifications of the embedded library in such a way that they are covered by the existing MPL-2.0 licensing statements. If you add new source code files to the library itself, insert a header containing your copyright line and an MPL-2.0 adequate licensing the statement.
- [**voluntary:**] Arrange your source code distribution so that the licensing elements (especially the MPL-2.0 license text and the *licensing files*) clearly refer only to the embedded library and do not affect the licensing of your own overarching work. It's a good tradition to keep embedded components like libraries, modules, snippets, or plugins in separate directories, which contain also all additional licensing elements.
- [**voluntary:**] Create a *modification text file*, if such a notice file still does not exist. *Add* a general description of your modifications to the *modification text file*. Incorporate the file into your distribution package.
- [**voluntary:**] Mark all modifications of the source code thoroughly, preferably in the modified source itself.
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing *copyright notice text files,* the name of the embedded MPL-2.0 licensed component, a link to its homepage, and a link to the MPL-2.0 license.

prohibits ...

- to remove or to alter any license elements (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the software package you have received.
- to promote any of your services based on the this software by trademarks, service marks, or logos linked to this MPL-2.0 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

[833] For implementing the handover of files correctly → OSLiC, p. 127

6.11.9 MPL-2.0-C9: Passing a modified library as embedded binary

means that you received an MPL-2.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10B[834]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the licensing elements (especially all copyright notices, patent notices, disclaimers of warranty, or limitations of liability) are retained in your package in exactly the form that you have received.If you compile the binary from the sources, ensure that all these licensing elements are also incorporated into the package.
- [**mandatory:**] Make the source code of the embedded library accessible via a repository under your own control: Push the source code package into the repository and make it downloadable via the Internet. Do no charge any fees from the user for downloading the source. Ensure, that this repository is online for a reasonable period of time after you ceased distributing the software.
- [**mandatory:**] Insert an easy to find description into the distribution package that explains how and where the code can be retrieved.
- [**mandatory:**] Execute the to-do list of use case MPL-2.0-C8 for the source code that you publish.[835]
- [**mandatory:**] Organize your modifications of the embedded library in such a way that they are covered by the existing MPL-2.0 licensing statements.
- [**voluntary:**] Create a *modification text file*, if such a notice file still does not exist. *Add* a general description of your modifications to the *modification text file*. Incorporate the file into your distribution package.
- [**voluntary:**] Give the recipient a copy of the MPL-2.0 license. If it is not already part of the software package, add it. If the licensing statement in the licensing file of the package does still not clearly state that the package is licensed under the MPL-2.0, additionally insert your own correct MPL-2.0 licensing file containing the sentence: *This*

[834] For details → OSLiC, pp. 126

[835] Making the code accessible via a repository means distributing the software in the form of source code. Hence, you must also fulfill all tasks of the corresponding use case.

Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.[836]

- [**voluntary:**] Arrange your binary distribution so that the licensing elements (especially the MPL-2.0 license text and the *licensing files*) clearly refer only to the embedded library and do not affect the licensing of your own overarching work. It's a good tradition to keep embedded components like libraries, modules, snippets, or plugins in separate directories, which contain also all additional licensing elements.
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also reproduce the content of the existing *copyright notice text files,* the name of the embedded MPL-2.0 licensed component, a link to its homepage, and a link to the MPL-2.0 license.

prohibits ...

- to remove or to alter any license elements (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the software package you have received.
- to promote any of your services based on the this software by trademarks, service marks, or logos linked to this MPL-2.0 software, except as required for reasonable and customary use in describing the origin of the software and reproducing the copyright notice.

6.11.10 Discussions and Explanations

The MPL-2.0 offers a section "Responsibilities" which contains nearly all requirements.[837] Only for some subordinate aspects, one has also to reflect other paragraphs.[838]§3 - concerning the trademarks With respect to this structure, we can detect the following tasks:

- In a more general attitude, the MPL-2.0 states that it "[...] does not grant any rights in the trademarks, service marks, or logos of any Contributor"—except as it may be necessary "to comply with" other requirements of the license.[839] The OSLiC rewrites the message as the interdiction to promote own services and products by and with such elements.
- The MPL-2.0 also generally prescribes that "you may not remove or alter the substance of any license notice (including copyright notices, patent notices,

836) For implementing the handover of files correctly → OSLiC, p. 127
837) cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp §3.
838) cf. id., l.c., wp pars pro to cf..
839) cf. id., l.c., wp §2.3.

disclaimer of warranties, or limitations of liabiliy) contained within the Source Code Form [...]"[840] This focussing to the "substance of any license notice" refers to the allowance to "[...] alter any license notices to the extent required to remedy known factual innacuracies".[841] Following its principle to offer one reliable way and to ignore variants of secondary importance, the OSLiC simplifies this condition to the general proscription to modify any licensing material for all use cases [MPL-2.0-C1 – MPL-2.0-C9]. But for emphasizing that this is a job which must be activily done, the OSLiC additionally rewrites this interdiction into all *2others* use cases [MPL-2.0-C2 – MPL-2.0-C9] as the task to retain the licensing elements in the form one has obtained them.

- Moreover, the MPL-2.0 requires for all "distributions of [the] source [code] form" that all modifications of the software "[...] must be under the terms of (the MPL-2.0)" and that the distributor "[...] must inform" all "recipients" that the software "[...] is governed by the terms of (the MPL-2.0), and how (the recipients) can obtain a copy of this license".[842] For the respective use case (MPL-2.0-C2, MPL-2.0-C4, MPL-2.0-C6, MPL-2.0-C8), the OSLiC rewrites these conditions so that each MPL-2.0 source code package must neccessarily contain the MPL-2.0 itself as textfile and an additional licensing file or statement strictly following the text given by the addendum of the MPL-2.0.[843] Because the MPL-2.0 is only a license with weak copyleft, the OSLiC proposes to separate the MPL-2.0 licensed, embedded component from the enclosing program (MPL-2.0-C8).
- But the MPL-2.0 does not explicitly require marking all modifications. Nevertheless, this is state of the art in computer emgineering. Therefore, with respect to the cases of distributing modified source code (MPL-2.0-C4, MPL-2.0-C6 and MPL-2.0-C8), the OSLiC proposes to mark all modifications inside of the source code and to update the description of the functional changes. In case of distributing the modified software in the form of binaries, it should be sufficient to describe the modifications only on the functional level.
- Furthermore, the MPL-2.0 requires that the "Covered Software"—in all cases of distributing it in an "Executable Form" (MPL-2.0-C3, MPL-2.0-C5, MPL-2.0-C7, MPL-2.0-C9)—"[...] must also be made available in Source Code Form [...]" and that the distributor "[...] must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost

840) cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp §3.4.
841) cf. id., ibid.
842) cf. id., l.c., wp §3.1.
843) cf. id., l.c., wp Exhibit A.

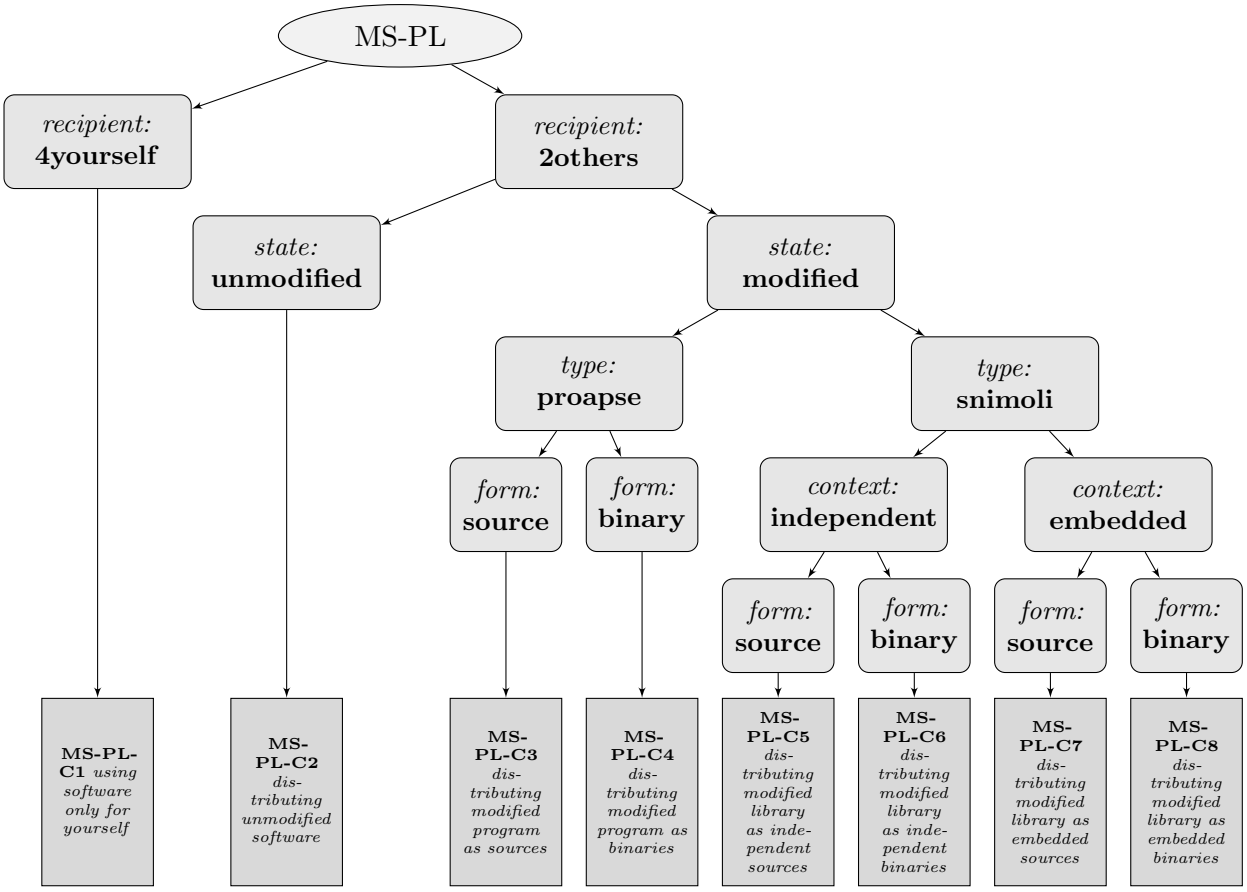
of distribution to the recipient".[844] The OSLiC rewrites these conditions as the obligation to offer a download service at no charge and to point towards this services inside of the distributed package.

- In this context, the MPL-2.0 allows to distribute the binaries under terms of another license "[...] provided that that the license for the Executable Form does not attempt to limit or alter the recipients' rights in the Source Code Form under this License."[845] This possibility might become important for those cases where the license compatibility must explicitly be managed. Normally, it should be sufficient also to distribute the binaries under the MPL-2.0. Thus, in case of distributing binaries (MPL-2.0-C3, MPL-2.0-C5, MPL-2.0-C7, MPL-2.0-C9), the OSLiC proposes to insert into the distribution packages the MPL-2.0 itself and an additional licensing file or statement strictly following the text given by the addendum of the MPL-2.0.[846] But again, because the MPL-2.0 is only a license with weak copyleft, the OSLiC proposes to separate the MPL-2.0 licensed embedded component from the overarching program (MPL-2.0-C9).
- Finally, one clearly has to state that the distribution of the source code required by the previous rule must, of course, follow the rules of distributing the software. Thus, the OSLiC requires in all cases of a binary distribution to execute also the task-lists of the respective source code use cases.

6.12 Microsoft Public License

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

844) cf. *Open Source Initiative*: The MPL-2.0 License (OSI), 2013, wp §3.2.a.
845) cf. id., l.c., wp §3.2.b.
846) cf. id., l.c., wp Exhibit A.



6.12.1 MS-PL-C1: Using the software only for yourself

means that you received MS-PL licensed software, that you will use it only for yourself and that you do not hand it over to any 3rd party in any sense.

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L, and OSUC-09N[847]

requires no tasks in order to fulfill the conditions of the Microsoft Public License with respect to this use case:

- 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 give the software to 3rd parties.

prohibits ...

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

[847] For details see pp. 112 - 124

6.12.2 MS-PL-C2: Passing the unmodified software

means that you received MS-PL licensed software which you are now going to distribute to third parties in the form of unmodified binaries or as unmodifed source code files. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent package.

covers OSUC-02S, OSUC-02B, OSUC-05S, OSUC-05B, OSUC-07S, OSUC-07B[848]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that all licensing elements (particularly all copyright, patent, trademark, and attribution notices that are part of 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 package.[849]
- [**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).

6.12.3 MS-PL-C3: Passing a modified program as source code

means that you received an MS-PL licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

covers OSUC-04S[850]

requires the following tasks in order to fulfill the license conditions:

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

[848] For details → OSLiC, pp. 113 – 121
[849] → OSLiC, p. 297
[850] For details → OSLiC, pp. 116

- [**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 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 to mention 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).

6.12.4 MS-PL-C4: Passing a modified program as binary

means that you received an MS-PL licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package.

covers OSUC-04B[851]

requires the following tasks in order to fulfill the license conditions:

- [**voluntary:**] Mark your modifications in the source code even if you do not intend to distribute it.
- [**voluntary:**] It's a good tradition to let the documentation of your distribution 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

[851] For details → OSLiC, pp. 116

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 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).

6.12.5 MS-PL-C5: Passing a modified library independently as source code

means that you received an MS-PL licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

covers OSUC-08S[852]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that all licensing elements (particularly all copyright, patent, trademark, and attribution notices that are part of 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 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).

[852] For details → OSLiC, pp. 122

6.12.6 MS-PL-C6: Passing a modified library independently as binary

means that you received an MS-PL licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers OSUC-08B[853]

requires the following 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 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).

6.12.7 MS-PL-C7: Passing a modified library as embedded source code

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

covers OSUC-10S[854]

requires the following tasks in order to fulfill the license conditions:

- [**voluntary:**] Ensure that all licensing elements (particularly all copyright, patent, trademark, and attribution notices that are part of the version you received are completely retained in your package.
- [**voluntary:**] Incorporate a complete copy of the MS-PL license into your package.
- [**voluntary:**] If you do not want to publish your modifications or your overarching application under the MS-PL too, then cleanly separate

[853] For details → OSLiC, pp. 123
[854] For details → OSLiC, pp. 125

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 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 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).

6.12.8 MS-PL-C8: Passing a modified library as embedded binary

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

covers OSUC-10B[855]

requires the following 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).

[855] For details → OSLiC, pp. 126

prohibits ...

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

6.12.9 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"[856].

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 portion[857].

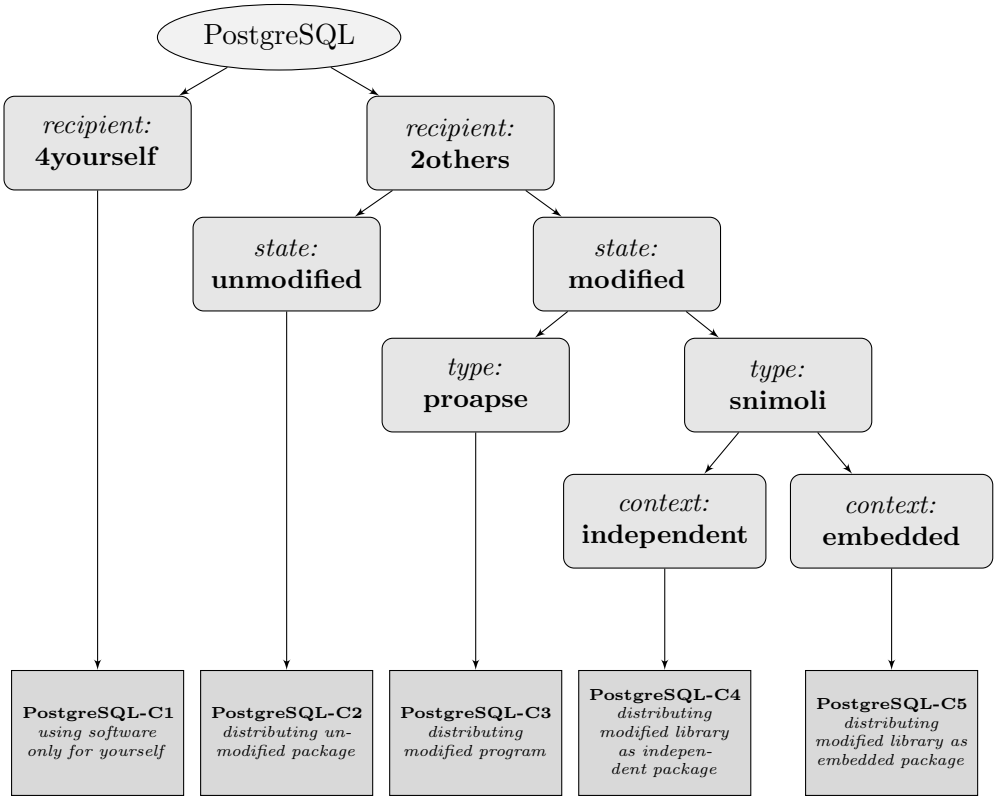
Based on these readings we decided ...

- ... 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[858] and the no warranty clause of the MS-PL by the OSLiC section concerning the power of the MS-PL[859] 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.

856) cf. *Open Source Initiative*: MS-PL, 2013, wp.
857) cf. id., l.c., wp. §3A, §3B, §3E.
858) → OSLiC, p. 61
859) → OSLiC, p. 44

6.13 PostgreSQL License

Like the MIT License Postgres License is a very permissive licenses. Thus, the PostgreSQL specific finder can be simplified:



6.13.1 PostgreSQL-C1: Using the software only for yourself

means that you received PostgreSQL licensed software, that you will use it only for yourself, and that you do not hand it over to any 3rd party in any sense.

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L, and OSUC-09N[860]

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

- You are allowed to use any kind of PostgreSQL licensed software in any sense and in any context without any other obligations if you do not give the software to third parties and if you do not modify the existing copyright notices or the existing permission notice.

prohibits nothing explicitely.

860) For details → OSLiC, pp. 112 - 124

6.13.2 PostgreSQL-C2: Passing the unmodified software

means that you received PostgreSQL licensed software which you are now going to distribute to third parties in the form of unmodified binaries or as unmodifed source code files. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent package.

covers OSUC-02S, OSUC-02B, OSUC-05S, OSUC-05B, OSUC-07S, OSUC-07B[861]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the complete PostgreSQL license including the copyright notice, the permission notices, and the PostgreSQL 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 or your additional material also contain a link to the original software (project) and its homepage.

prohibits nothing explicitely.

6.13.3 PostgreSQL-C3: Passing a modified program

means that you received a PostgreSQL licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form binaries or as source code files.

covers OSUC-04S, OSUC-04B[862]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the complete PostgreSQL license including the copyright notice, the permission notices, and the PostgreSQL disclaimer are retained in your package in the form you have received them.
- [**voluntary:**] Mark your modifications in the source code, regardless whether you want to distribute the code or not.
- [**voluntary:**] It's a good tradition to let the documentation of your distribution or your additional material also contain a link to the original software (project) and its homepage.

[861] For details → OSLiC, pp. 112 – 121
[862] For details → OSLiC, pp. 116

- [**voluntary:**] You can add information about your own work or modifications to an existing copyright notice presented by the program.
- [**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 based on a version originally licensed under the PostgreSQL license. Because you are already modifying the program, you may want to add such a hint, if the original copyright notice lacks such a statement.

prohibits nothing explicitely.

6.13.4 PostgreSQL-C4: Passing a modified library independently

means that you received a PostgreSQL licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binaries or as source code files together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component, regardless whether you distribute it in the form of binaries or as source code files.

covers OSUC-08S, OSUC-08B[863]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the complete PostgreSQL license including the copyright notice, the permission notices, and the PostgreSQL disclaimer are retained in your package in the form you have received them.
- [**voluntary:**] Mark your modifications in the source code, regardless whether you want to distribute the code or not.
- [**voluntary:**] It's a good tradition to let the documentation of your distribution or your additional material also contain a link to the original software (project) and its homepage.

prohibits nothing explicitely.

6.13.5 PostgreSQL-C5: Passing a modified library as embedded component

means that you received a PostgreSQL licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binaries or as source code files together with another larger software unit which contains

[863] For details → OSLiC, pp. 123

this code snippet, module, library, or plugin as an embedded component, regardless whether you distribute it in the form of binaries or as source code files.

covers OSUC-10S, OSUC-10B[864]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the complete PostgreSQL license including the copyright notice, the permission notices, and the PostgreSQL disclaimer are retained in your package in the form you have received them.
- [**voluntary:**] Mark your modifications in the source code, regardless whether you want to distribute the 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 PostgreSQL license. And it is a good tradition to insert links to the homepage or download page of this embedded component.
- [**voluntary:**] It's a good tradition to let the documentation of your distribution or your additional material also contain a link to the original software (project) and its homepage.
- [**voluntary:**] Arrange your distribution so that the original licensing elements (in particular the PostgreSQL license text containing the copyright notices of the original author(s), the permission notices and the PostgrSGL disclaimer) clearly refer only to the embedded library and do not affect the licensing of your own overarching work. Consider keeping embedded libraries, modules, snippets, or plugins in separate directories which also contain all their licensing elements.

prohibits nothing explicitely.

6.13.6 Discussions and Explanations

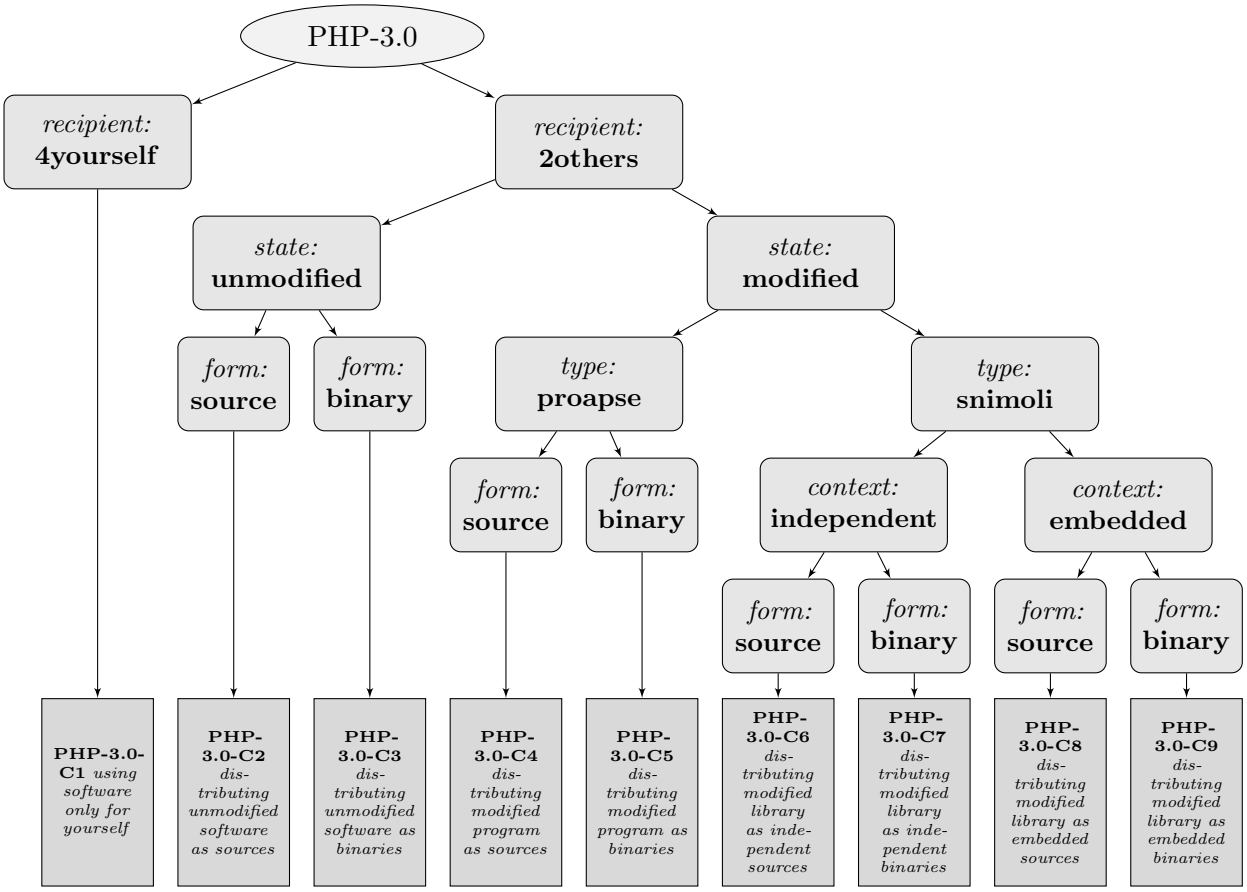
The PostgreSQL-License follows the structure of the MIT license: it, too, contains (1) a copyright notice, (2) a paragraph saying that you are allowed to do almost anything you want, followed (3) by the condition that the copyright notice, the permission notes, and the disclaimer "[...] apperar in all copies", and (4) the well known disclaimer.[865] Moreover, like the MIT license, the PostgreSQL does

[864] For details → OSLiC, pp. 125
[865] cf. *Open Source Initiative*: PostgreSQL License, 2013, wp.

not talk about the difference between source code and object code. So, you can apply the analysis of the MIT license[866] also to the PostgreSQL.

6.14 PHP-3.0 licensed software

The PHP-3.0 license contains a few more conditions than the MIT license and additionally distinguishes the "redistribution of source code"[867] from the "redistribution in binary form".[868] Nevertheless, the PHP-3.0 license focusses only on the redistribution or—as we call it in the OSLiC—*the 2others use cases*. Thus, the PHP-3.0 specific finder can be simplified:



6.14.1 PHP-3.0-C1: Using the software only for yourself

means that you received PHP-3.0 licensed software, that you will use it only for yourself, and that you do not hand it over to any third party in any sense.

866) → OSLiC, p. 275
867) cf. *Open Source Initiative*: PHP-3.0, 2013, wp.
868) cf. id., ibid.

covers OSUC-01, OSUC-03L, OSUC-03N, OSUC-06L, OSUC-06N, OSUC-09L, and OSUC-09N[869]

requires no tasks in order to fulfill the conditions of the PHP 3.0 License with respect to this use case:

- You are allowed to use any kind of PHP-3.0 software in any sense and in any context without any obligations as long as you do not give the software to third parties.

prohibits ...

- to endorse or promote any service you establish based on this software by the name 'PHP.'

6.14.2 PHP-3.0-C2: Passing the unmodified software as source code

means that you received PHP-3.0 licensed software which you are now going to distribute to third parties in the form of unmodified source code files or as unmodified source code package. In this case it makes no difference if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or as an embedded unit.

covers OSUC-02S, OSUC-05S, OSUC-07S[870]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the complete PHP-3.0 license (especially the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer) are retained in your package in the form you have received them.
- [**mandatory:**] Let the documentation of your distribution or your additional material also contain a line of acknowledgment in the form: "This product includes PHP, freely available from http://www.php.net/"
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer.

prohibits ...

- to endorse or promote your product by mentioning PHP, especially not by making the string 'PHP' part of its name.

869) For details → OSLiC, pp. 112 – 124
870) For details → OSLiC, pp. 112 – 120

6.14.3 PHP-3.0-C3: Passing the unmodified software as binary

means that you received PHP-3.0 licensed software which you are now going to distribute to third parties in the form of unmodified binary files or as unmodified binary package. In this case it does not matter if you distribute a program, an application, a server, a snippet, a module, a library, or a plugin as an independent or an embedded unit.

covers OSUC-02B, OSUC-05B, OSUC-07B[871]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the complete PHP-3.0 license (especially the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer) are *reproduced* by your package in the form you have received them. If you compile the binary file from the source code package and if this process does not also generate and integrate the licensing files then create the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer in the form present in the source code package and insert these files into your distribution manually.[872][873]
- [**mandatory:**] Let the documentation of your distribution or your additional material also contain a line of acknowledgment in the form: "This product includes PHP, freely available from http://www.php.net/"
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer.

prohibits ...

- to endorse or promote your product by mentioning PHP, especially not by making the string 'PHP' part of its name.

6.14.4 PHP-3.0-C4: Passing a modified program as source code

means that you received a PHP-3.0 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package.

covers OSUC-04S[874]

871) For details → OSLiC, pp. 113 – 121
872) Because you are distributing an unmodified binary, you could assume that the copright screens of the application do already what they have to do.
873) For implementing the handover of files correctly → OSLiC, p. 127
874) For details → OSLiC, pp. 116

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the complete PHP-3.0 license (especially the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer) are retained in your package in the form you have received them.
- [**mandatory:**] Let the documentation of your distribution or your additional material also contain a line of acknowledgment in the form: "This product includes PHP, freely available from http://www.php.net/"
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the PHP-3.0 conditions, and the PHP-3.0 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 PHP-3.0 license.. Because you are already modifying the program you can also add such a hint if the original copyright notice lacks such a statement. If such a notice is missing in the copyright screen, consider, if it is possible to let it *reproduce* the complete PHP-3.0 license including the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer (as it is required for binary distributions.)[875]
- [**voluntary:**] Mark your modifications in the source code.

prohibits ...

- to endorse or promote your product by mentioning PHP, especially not by making the string 'PHP' part of its name.

6.14.5 PHP-3.0-C5: Passing a modified program as binary

means that you received a PHP-3.0 licensed program, application, or server (proapse), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package.

covers OSUC-04B[876]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Let the documentation of your distribution or your additional material also contain a line of acknowledgment in the form: "This product includes PHP, freely available from http://www.php.net/"

[875] Following distributors of compiled versions will appreciate your prepatory work.
[876] For details → OSLiC, pp. 116

- [**mandatory:**] Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer.
- [**voluntary:**] Ensure that the complete PHP-3.0 license (especially the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer) are *reproduced* by your package in the form you have received them. If you compile the binary file from the source code package and if this process does not also generate and integrate the licensing files then create the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer in the form present in the source code package and insert these files into your distribution manually.
- [**voluntary:**] Mark your modifications in the source code, even if you do not want to distribute the code.

prohibits ...

- to endorse or promote your product by mentioning PHP, especially not by making the string 'PHP' part of its name.

6.14.6 PHP-3.0-C6: Passing a modified library as independent source code

means that you received a PHP-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of source code files or as a source code package, but without embedding it into another larger software unit.

covers OSUC-08S[877]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the complete PHP-3.0 license (especially the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer) are retained in your package in the form you have received them.
- [**mandatory:**] Let the documentation of your distribution or your additional material also contain a line of acknowledgment in the form: "This product includes PHP, freely available from http://www.php.net/"
- [**voluntary:**] Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer.
- [**voluntary:**] Mark your modifications in the source code.

[877] For details → OSLiC, pp. 122

prohibits ...

- to endorse or promote your product by mentioning PHP, especially not by making the string 'PHP' part of its name.

6.14.7 PHP-3.0-C7: Passing a modified library as independent binary

means that you received a PHP-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package but without embedding it into another larger software unit.

covers OSUC-08B[878]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Let the documentation of your distribution or your additional material also contain a line of acknowledgment in the form: "This product includes PHP, freely available from http://www.php.net/"
- [**mandatory:**] Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer.
- [**voluntary:**] Ensure that the complete PHP-3.0 license (especially the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer) are *reproduced* by your package in the form you have received them. If you compile the binary file from the source code package and if this process does not also generate and integrate the licensing files then create the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer in the form present in the source code package and insert these files into your distribution manually.
- [**voluntary:**] Mark your modifications in the source code, even if you do not want to distribute the code.

prohibits ...

- to endorse or promote your product by mentioning PHP, especially not by making the string 'PHP' part of its name.

6.14.8 PHP-3.0-C8: Passing a modified library as embedded source code

means that you received a PHP-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to

[878] For details → OSLiC, pp. 123

distribute this modified version to third parties in the form of source code files or as a source code package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10S[879]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Ensure that the complete PHP-3.0 license (especially the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer) are retained in your package in the form you have received them.
- [**mandatory:**] Let the documentation of your distribution or your additional material also contain a line of acknowledgment in the form: "This product includes PHP, freely available from http://www.php.net/"
- [**mandatory:**] Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the PHP-3.0 conditions, and the PHP-3.0 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 uses a component licensed under the PHP-3.0 license. So, let the copyright screen of the enclosing program *reproduce* the complete PHP-3.0 license including the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer (as it is required for binary distributions.)[880]
- [**voluntary:**] Mark your modifications in the source code.
- [**voluntary:**] Arrange your source code distribution so that the licensing elements (especially the PHP-3.0 license text, the specific copyright notice of the original author(s), and the PHP-3.0 disclaimer) clearly refer only to the embedded library and do not affect the licensing of your own overarching work. It's a good tradition to keep embedded components like libraries, modules, snippets, or plugins in separate directories, which contain also all additional licensing elements.

prohibits ...

- to endorse or promote your product by mentioning PHP, especially not by making the string 'PHP' part of its name.

[879] For details → OSLiC, pp. 125

[880] Following distributors of compiled versions will appreciate your prepatory work.

6.14.9 PHP-3.0-C9: Passing a modified library as embedded binary

means that you received a PHP-3.0 licensed code snippet, module, library, or plugin (snimoli), that you modified it, and that you are now going to distribute this modified version to third parties in the form of binary files or as a binary package together with another larger software unit which contains this code snippet, module, library, or plugin as an embedded component.

covers OSUC-10B[881]

requires the following tasks in order to fulfill the license conditions:

- [**mandatory:**] Let the documentation of your distribution or your additional material also contain a line of acknowledgment in the form: "This product includes PHP, freely available from http://www.php.net/"
- [**mandatory:**] Let the documentation of your distribution and/or your additional material also contain the original copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer.
- [**voluntary:**] Ensure that the complete PHP-3.0 license (especially the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer) are *reproduced* by your package in the form you have received them. If you compile the binary file from the source code package and if this process does not also generate and integrate the licensing files then create the copyright notice, the PHP-3.0 conditions, and the PHP-3.0 disclaimer in the form present in the source code package and insert these files into your distribution manually.
- [**voluntary:**] Mark your modifications in the source code, even if you do not want to distribute the code.
- [**voluntary:**] Arrange your binary distribution so that the licensing elements (especially the PHP-3.0 license text, the specific copyright notice of the original author(s), and the PHP-3.0 disclaimer) clearly refer only to the embedded library and do not affect the licensing of your own overarching work. It's a good tradition to keep embedded components like libraries, modules, snippets, or plugins in separate directories, which contain also all additional licensing elements.

prohibits ...

- to endorse or promote your product by mentioning PHP, especially not by making the string 'PHP' part of its name.

[881] For details → OSLiC, pp. 126

6.14.10 Discussions and Explanations

First of all, it might surprise some readers that the OSLiC also describes the open source use cases which concern the distribution of binary files although it deals with the PHP-3.0 license. PHP is a script language. Thus, delivering the source code seems to be a must. But one has to consider that the PHP-3.0 license could also be applied to works which are based on other languages constituted on the compiler paradigm. Or there might a PHP compiler be used.

It might also surprise some readers that in case of the binary distribution of modifications the condition to repoduce the php license in the documentation is a *must,* while its reproduction in a copyright screen of the program is a *should.* This is directly caused by the binary-condition of the php license which expressly requires that "Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution."[882] But of course, implementing the *must* and the *should* is best.

[882] cf. *Open Source Initiative*: PHP-3.0, 2013, wp.

7 Conclusion

During the last 4 years, we have developed this **O**pen **S**ource **Li**cense **C**ompendium. We had the honor and the pleasure to discuss our ideas with many open source experts, for example with those, who visit the European Legal and Licensing Workshop, organized by the FSFE. We were invited to present our work on different conferences, in Germany, in Europe, and even in Asia. We got a very encouring feedback. Today we know what we only supposed when we started: We could indeed close an important gap by offering a simple and reliable way to ascertain what one has to do for using open source software compliantly. We are proud of having gone this long way. And we pride ourselves on the fact that – today – the OSLiC is officially listed by the OSI as one of those tools by which one can manage the open source compliance[883].

But, we also got adjusting feedback: Namely our initial premise was justifiably not really accepted by the community. We were told that the software developers themselves would never use our OSLiC. They would never read a book of more than 300 pages full of lists and tables – as long as this book was not a specification of a computer language. The OSLiC would be too large and too complex for simplifying the daily life of the open source users. It would be an excellent foundation for becoming an open source license expert – but not a tool for the desk. And indeed, it was simply silly to assume that software developers, project managers, or IT managers can directly understand and use the OSLiC: reading the OSLiC the first time has a discouraging shock effect. Today, also we know this.

Nevertheless, it was very important for us to fall for the charme of this illusion. Without this error, we never would have started the development of the OSLiC. And thus, we never would have find the idea to organize the issue in form of finders and a 5 question form. Without this error, we today would never have a work which justifies and proves each single assertion by quoting the licenses and the experts. And without this frightening feedback we received, we never would have got one of our best and encouraging experiences:

When we had accepted the feedback, we directly decided to develop an online version of the OSLiC, the Open Source Compliance Advisor, also know as OSCAd[884]. We distributed it under the terms of the AGPL. Then, the company Amadeus decided to take over the development of this online tool. We, on our

883) → http://osi.xwiki.com/bin/Projects/Process+and+Compliance+Resources
884) → http://opensource.telekom.net/oscad/

side, inserted an export interface into the OSLiC. They, on their side, rewrote the OSCAd and integrated an import interface. So – finally – we both were able to focus on only one specific aspect: they took the responsibility for computing and maintaining the online tool[885], we took the responsibility maintaining for the fundamental analysis of the open source licenses[886].

Thus, we concretely experienced the advantages of sharing ideas and sources, which were so often emphazied: Playing the open source game actively means giving a bit and getting back a lot. Playing the open source game actively means saving the own resources.

Therefore, you may also take the fact that we finally could indeed publish the version 1.0 of the OSLiC as a thankful profound curtsey to the open source community!

[885] → https://github.com/AmadeusITGroup/oscad
[886] → https://github.com/dtag-dbu/oslic/

8 Appendices

8.1 Some Additional Remarks on the OSLiC Quotation Style

We have already characterized the general tone of our footnotes[887]. 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

887) → p. 14

all these e-wares[888] we read[889]. 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 data[890]:

- 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 url[891] and the date when we downloaded / saw the text.

8.2 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 myths[892] we met:

BE CAREFUL: THIS SECTION MUST THOROUGHLY BE REVIEWED AND REWRITTEN. IT'S ONLY AN OUTLINE!!! Do not quote part of it. It must be verified.

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

889) 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.

890) 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.

891) 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.

892) 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.

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.

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 modfied (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 you give the binary to. And if you only give your improvement only one person or a group of persons, 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. This limitation is based on the open source definition which clearly states that a license – which wants to become an open source license – "shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources" and that the license under this circumstances "[...] shall not require a royalty or other fee for such sale"[893]. If you combine this constraint with the requirements that an open source license "[...] must not restrict anyone from making use of the program [...]"[894] and that it "[...] must allow distribution in source code as well as compiled form [...]"[895], you can generally conclude that none of the open source licenses may require a fee for using and/or distributing the program. But being paid for the service to install the program, to collect and compile a customer specific version, and/or to monitor the environment is of course not excluded by this condition.

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 include GNU software. Then, in 1983? Ian Murdock was supported by RMS and its FSF to build a really free distribution (Debian) containg 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 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

893) cf. *Open Source Initiative*: The Open Source Definition, 2012, §1.
894) cf. id., l.c., §6.
895) cf. id., l.c., §2.

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 derivated from these needs.

8.2.1 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.
- 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 incorrect[896].

[896] 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.

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.

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 application[897]. 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 this[898].

[897] As general examples let us mention Palamida (`http://www.palamida.com/`) and BlackDuck (`http://www.blackducksoftware.com/`).

[898] 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.

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 releated 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 open source software[899] with a specific license[900]. 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*[901]. 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 together[902]. 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 details[903], does not mention anything about the license which allows

899) 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]

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

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

902) 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.

903) cf. *Coar, Ken* a. *Rich Bowen*: Apache Kochbuch; deutsche Übersetzung v. Jochen

for installation, configuration and maintenance. Neither the index lists the word 'license'[904], nor the chapters 'Installation'[905] or the chapter 'Miscellaneous'[906] 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 it[907]. 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, 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 difficult[908]. So again: Can you find your answer in another book, which is *Amazon's* current top recommendation for the search term *'apache server'*[909]? Not really: Sascha Kersken's Apache 2.2 Handbook offers a license chapter, but it is only two pages long[910]. 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 improvements[911].

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*[912], founded in 2000 as a private institute to track the phenomenon 'free software' from the viewpoint of (German) lawyers[913]. Besides

Wiedmann; Beijing [...]: O'Reilly, 2004, ISBN 3–89721–371–0, et passim.

904) cf. *Coar a. Bowen*: Apache Kochbuch, 2004, pp. 245ff, esp. p. 250.

905) cf. id., l.c., pp. 1ff.

906) cf. id., l.c., pp. 219ff.

907) 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 ".

908) 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?

909) Tested on `http://www.amazon.de/` at 2011-08-31.

910) 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.

911) cf. id., l.c., p. 112.

912) originally: "Institut für Rechtsfragen der Freien und open source software". Main entry point for its site is the URL `http://www.ifross.org/`.

913) cf. *ifross*: Ziele, Aufgaben, Geschichte; 2011 ⟨URL: `http://www.ifross.org/node/16`⟩ – reference download: 2011-09-05, wp.

many other aspects this site offers a very well and thoroughly elaborated FAQ[914] 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)[915].

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 bibliographies[916]: the book of Jaeger and Metzger which concerns – loosely translated – *the judicial framework requirement for open source software*[917]. 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 duties[918] of the software user[919]. Many other important aspects of the topic *open source* are discussed, too[920].

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 edition[921] and in the second edition[922] nor in the recently published third edition[923]. 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

914) cf. *ifross*: FAQ; 2011 ⟨URL: http://www.ifross.org/faq-haeufig-gestellte-fragen⟩ – reference download: 2011-09-05, wp.
915) cf. *ifross*: ifross Lizenz-Center, 2011, wp.
916) at least in that German judicial literature dealing with open source
917) 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).
918) cf. id., l.c., pp. 30ff.
919) For getting a good survey of the structure and the line of thought see the contents cf. id., l.c., pp. VIIIf.
920) pars pro toto: have a look at the chapter concerning the liability: cf. id., l.c., pp. 137ff.
921) cf. id., l.c., pp. VIff.
922) 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. VIIff.
923) 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.

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'[924]. It aims to be a 'pragmatic guidebook' and an 'assistance' for 'publishing software under the conditions of an open source license'[925]. 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 source software[926].

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 public[927]. 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 author[928]. 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 donation[929]. And last but not least this article actually summarizes the idea of Copyleft and the differences between LGPL and GPL[930].

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.

---

924) approximately translated

925) cf. *Bretschneider, Ulrich*, *Rainer Glaschick*, a. *Gernot Gräfe*: Ratgeber für die Veröffentlichung von Open-Source-Software durch eine Hochschule; In *Asche* et al.: Open Source. Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen, 2008, pp. 166f (originally: ein "pragmatischer Ratgeber" zur "Veröffentlichung einer Software unter den Rahmenbedingungen einer Open-Source-Lizenz").

926) cf. id., l.c., pp. 186 (originally: ein "Ratgeber", der es erlaubt " (...) die zu berücksichtigende Aspekte (strukturiert abzuarbeiten) (...) " und einen "rechtlich nicht angreifbaren Veröffentlichungsprozess" zu ermöglichen).

927) cf. id., l.c., pp. 170 and 181.

928) cf. id., l.c., p. 173.

929) cf. id., l.c., p. 177.

930) cf. id., l.c., p. 181.

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 software[931]. 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 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

931) cf. *Bretschneider, Glaschick, a. Gräfe*: Ratgeber für die Veröffentlichung von Open-Source-Software durch eine Hochschule, 2008a, p. 186.

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 of these licenses would not be valid with respect to a legal system[932], we wanted to take them as our guideline – at least while they do not violate more general laws[933]. 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 anything[934], we would interpret the open issues

[932] 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.

[933] what they clearly do not do!

[934] 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

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 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.

8.2.2 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*[935], the *open source software movement*[936], or the GNU-Project[937]. 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

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.

935) 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.

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

937) 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.

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 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 the 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 derivated 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 Distrobution (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]

[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 Qualilty 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

Perspectives on free and open source software; o.O.?: ???, 2005

Proceedings of the Linux Symposium; Ottawa, 2006

Ågerfalk, Pär et al., editors: Open Source Software: New Horizons; 6th International IFIP WG 2.13 Conference on Open Source Systems, OSS 2010; (= IFIP Advances in Information and Communication Technology, [Vol./No.] 319) Berlin, Heidelberg u. New York: Springer, 2010, BibWeb/PDF, ISBN 978–3–642–13243–8

Ahtiainen, Aleksi, *Sami Surakka*, a. *Mikko Rahikainen*: Plaggie: GNU-licensed Source Code Plagiarism Detection Engine for Java Exercises; in: Proceedings of the 6th Baltic Sea Conference on Computing Education Research; New York, NY, USA: ACM, 2006 (= Baltic Sea '06) ⟨URL: http://doi.acm.org/10.1145/1315803.1315831⟩ – reference download: 2011-12-29, BibWeb/PDF, pp. 141–142

Airlie, D. M.: Open Source Graphic Drivers. They Don't Kill the Kittens; In Proceedings of the Linux Symposium, 2006, p. 19.26

Alexi, O. a. *J. Henkel*: Promoting the Penguin: Who Is Advocating Open Source Software in Commercial Settings? München, 2007 ⟨URL: http://ssrn.com/abstract/=988363⟩

Alexy, Oliver: Free Revealing. How Firms Can Profit From Being Open; Wiesbaden: Gabler, 2009 (= Gabler Edition Wissenschaft), Print and BibWeb/PDF, ISBN 978–3–8349–1475–0

Allman, Eric: A Conversation with Chris DiBona; in: Queue, 1 July (2003), pp. 10–19 ⟨URL: http://doi.acm.org/10.1145/945074.945130⟩ – reference download: 2011-12-29, BibWeb/PDF

Allman, Eric a. *Marshall Kirk McKusick*: From the Editors: Open Source Revisited; in: Queue, 2 May (2004), pp. 8–9 ⟨URL: http://doi.acm.org/10.1145/1005062.1005072⟩

Alspaugh, Thomas A., *Hazeline U. Asuncion*, a. *Walt Scacchi*: Analyzing software licenses in open architecture software systems; In Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009, pp. 54–57 ⟨URL: http://dx.doi.org/10.1109/FLOSS.2009.5071361⟩ – reference download: 2011-12-28, BibWeb/PDF

Alspaugh, Thomas A., *Walt Scacchi*, a. *Hazeline U. Asuncion*: Software Licenses in Context: The Challenge of Heterogeneously-Licensed Systems; in: JAIS, 11 (2010), No. 11/12, pp. 730–755, BibWeb/PDF

Amega-Selorm, Charles a. *Johanna Awotwi*: Free and Open Source Software (FOSS): It's Significance or Otherwise to the E-Governance Process in Ghana; in: Proceedings of the 4th International Conference on Theory and Practice of Electronic Governance; New York, NY, USA: ACM, 2010 (= ICEGOV '10) ⟨URL: http://doi.acm.org/10.1145/1930321.1930342⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0058–2, pp. 91–95

anonymous: Mircosoft und Open Source: nichts als Ärger; 2000 ⟨URL: http://www.heise.de/newsticker/meldungen/8314⟩

anonymous: Microsoft und die GPL: Freiheit, die ich meine ... 2002 ⟨URL: http://www.heise.de/newsticker/meldung/26355⟩

anonymous: Mircosoft Shared Source Initiative Overview; 2004 ⟨URL: http://www.microsoft.com/resources/sharedsource/Intiative/Intiative.mspx⟩

Anonymous: The Triumph of the Commons; in: The Economist (2005)

Anvaari, Mohsen a. *Slinger Jansen*: Evaluating Architectural Openness in Mobile Software Platforms; in: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume; New York, NY, USA: ACM, 2010 (= ECSA '10) ⟨URL: http://doi.acm.org/10.1145/1842752.1842775⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0179–4, pp. 85–92

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

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

Arlt, Brinkel, a. *Volkmann; Spindler, Gerald, editor*: 'BSD' - und 'Mozilla'-artige Lizenzen; In *Spindler*: Rechtsfragen bei Open Source Software, 2004, pp. 317–372, Print

Arnö, Kaj: Dual Licensing - A Business Model from the Second Generation of Open-Source Companies; In *Wynants a. Cornelius*: How Open is the Future?, 2005, pp. 479–486

Asay, Matt: A Funny Thing Happened on the Way to the Market: Linux, the General Public License, and a New Model for Software Innovation; Stanford CA, 2002, Web/Pdf ⟨URL: http://www.linuxdevices.com/files/misc/asay-paper.pdf⟩

Asche, Michael et al., editors: Open Source. Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen; (= POWeR / Patent Offensive Westfalen Ruhr, [Vol./No.] 3) Münster, New York, München [... etc.]: Waxmann, 2008, Print, ISBN 978–3–8309–1845–5

Ascher, David: Is Open Source Right for You? in: Queue, 2 May (2004), pp. 32–38 ⟨URL: http://doi.acm.org/10.1145/1005062.1005065⟩ – reference download: 2011-12-28, BibWeb/PDF

Asiri, Sami: Open Source Software; in: SIGCAS, 33 March (2003), p. 2 ⟨URL: http://doi.acm.org/10.1145/966498.966501⟩ – reference download: 2011-12-28, BIbWeb/HTML

Asundi, Jai: The Need for Effort Estimation Models for Open Source Software Projects; In Proceedings of the Fifth Workshop on Open Source Software Engineering, 2005, pp. 6:1–6:3 ⟨URL: http://doi.acm.org/10.1145/1082983.1083260⟩ – reference download: 2011-12-29, BibWeb/PDF

Axel Metzger, Till Jaeger und: Die neue Version 3 der GNU Genereal Public License; in: GRUR, o.A. (2008), No. 2, pp. 130–137, Copy

Ayala, Claudia et al.: Challenges of the Open Source Component Marketplace in the Industry; conference contribution; In *Boldyreff et al.*: Open Source Ecosystems, 2009, pp. 213–224, BibWeb/PDF

Azzi, R. Michael: CPR: How Jacobsen V. Katzer Resuscitated the Open Source Movement; in: University of Illinois Law Review, (2010), No. 4, pp. 1271–1302, BibWeb/PDF

Babcock, Charles: Big Test For Open Source GPL; in: Informationweek, 17 December (2006), p. np., Copy

Bach, Paula M. a. *John M. Carroll*: Characterizing the Dynamics of Open User Experience Design: The Cases of Firefox and OpenOffice.org; in: JAIS, 11 (2010), No. 12, pp. 902–925, BibWeb/PDF

Backu, Frieder: Open Source Software und Interoperabilität; in: ITRB (IT-Rechtsberater), (2003), p. 180

Baerwolff, Matthias, Robert A. Gehring, a. *Bernd Lutterbeck, editors*: Open Source Jahrbuch 2005. Zwischen Softwareentwicklung und Gesellschaftsmodell; Berlin: Lehmanns Media, 2005 ⟨URL: http://www.opensourcejahrbuch.de/download/jb2005/OpenSourceJahrbuch2005_online.pdf⟩ – reference download: 2011-10-17, Print & FreeWeb/PDF, ISBN 3–86541–059–6

Bain, Malcolm et al.: Legal Aspects of the Information Society; Oberta de Catalunya: Free Technology Academy, 2010 ⟨URL: http://www.ftacademy.org/materials/fsm/6#1⟩ – reference download: 2012-101-20, FreeWeb/PDF

Baird, Stacy Avery: The Heterogeneous World of Proprietary and Open-Source Software; in: Proceedings of the 2nd international conference on Theory and practice of electronic governance; New York, NY, USA: ACM, 2008 (= ICEGOV '08) ⟨URL: http://doi.acm.org/10.1145/1509096.1509143⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 978–1–60558–386–0, pp. 232–238

Bake, Pio: Open Source Software, Competition and Potential Entry; Working Paper; 2003, Web/Pdf ⟨URL: http://www.berlecon.de/tw/osscompetition.pdf⟩

Bakker, Arno, *Maarten Van Steen*, a. *Andrew S. Tanenbaum*: A Wide-Area Distribution Network for Free Software; in: Transaction on Internet Technology, 6 August (2006), No. 3, pp. 259–281 ⟨URL: http://doi.acm.org/10.1145/1151087.1151089⟩ – reference download: 2011-12-29, BibWeb/PDF

Baldi, Stefan, *Hauke Heier*, a. *Anett Mehler-Bicher*: Open Courseware and Open Source Software; in: Communications of the ACM, 46 September (2003), No. 9, pp. 105–107 ⟨URL: http://doi.acm.org/10.1145/903893.903922⟩ – reference download: 2011-12-29, BibWeb/PDF

Barcellini, Flore, *Françoise Détienne*, a. *Jean Marie Burkhardt*: Cross-Participants: Fostering Design-Use Mediation in an Open Source Software Community; in: Proceedings of the 14th European Conference on Cognitive Ergonomics: Invent! Explore! New York, NY, USA: ACM, 2007 (= ECCE '07) ⟨URL: http://doi.acm.org/10.1145/1362550.1362564⟩, ISBN 978–1–84799–849–1, pp. 57–64

Barcellini, Flore et al.: Thematic Coherence and Quotation Practices in OSS Design-Oriented Online Discussions; in: Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work; New York, NY, USA: ACM, 2005 (= GROUP '05) ⟨URL: http://doi.acm.org/10.1145/1099203.1099237⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 1–59593–223–2, pp. 177–186

Bayersdorfer, Mitch: Managing a Project With Open Source Components; in: interactions, 14 November/December (2007), pp. 33–34 ⟨URL: http://doi.acm.org/10.1145/1300655.1300677⟩ – reference download: 2011-12-29, BibWeb/PDF

Baytiyeh, Hoda a. *Jay Pfaffman*: Open source software: A community of altruists; in: Computers in Human Behavior, 26 (2010), p. 1345–1354, BibWeb/PDF

Beard, Ashley a. *Hyunju Kim*: A Survey On Open Source Software Licenses; in: JCSC, 22 (2007), No. 4, pp. 205–211 ⟨URL: http://dl.acm.org/citation.cfm?id=1229637.1229673⟩

Behlendorf, Brian: Open Source as a Business Strategy; In *DiBona, Ockman, a. Stone*: Open Sources, 1999, pp. 149–170

Behlendorf, Brian: How Open Source Can Still Save the World; conference contribution; In *Boldyreff et al.*: Open Source Ecosystems, 2009, p. 2, BibWeb/PDF

Beigbeder, Michel, *Wray Buntine*, a. *Wai Gen Yee*: Open Source Search and Research; in: Proceedings of the 2006 International Workshop on Research Issues in Digital Libraries; New York, NY, USA: ACM, 2007 (= IWRIDL '06) ⟨URL: http://doi.acm.org/10.1145/1364742.1364748⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–59593–608–4, pp. 5:1–5:7

Berglund, Erik a. *Michael Priestley*: Open-Source Documentation: In Search of User-Driven, Just-in-Time Writing; in: Proceedings of the 19th annual international conference on Computer documentation; New York, NY, USA: ACM, 2001 (= SIGDOC '01) ⟨URL: http://doi.acm.org/10.1145/501516.501543⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–58113–295–6, pp. 132–141

Bergquist, Magnus a. *Jan Ljungberg*: The power of gifts: organizing social relationships in open source communities; in: Information Systems Journal, 11 (2001), pp. 305–320

Berlecon[-]Research: Basics of Open Source Software Markets and Business Models; FLOSS Final Report - Part 3; 2002, Web/Pdf ⟨URL: http://www.berlecon.de/studien/downloads/200207FLOSS_Basics.pdf⟩

Berlecon[-]Research: Firms Open Source Activities: Motivations and Policy Implications; FLOSS Final Report - Part 2; 2002, Web/Pdf ⟨URL: http://www.berlecon.de/studien/downloads/200207FLOSS_Activities.pdf⟩

Berry, David M.: Copy, Rip, Burn; The Politics of Copyleft and Open Source; London: Pluto Press, 2008, Print, ISBN 978–0–7453–2414–2

Bessen, James: What Good is Free Software? In *Hahn*: Government Policy toward Open Source Software, 2002, pp. 12–34

Bessen, James: Holdup and licensing of cumulative innovations with private information; in: Economics Letters, 82 (2004), No. 3, pp. 321–326

Bessen, James a. *Robert M. Hunt*: An empirical look at software patents; in: Federal Reserve Bank of Philadelphia, Working Paper 03-17 (2004)

Bezroukov, Nikoplai: BSD vs. GPL: Part 2: The Dynamic Properties of BSD and GPL Licenses in the Context of the Program Life Cycle; 2003 ⟨URL: http://www.softpanorama.org/Copyright/License-classification/social_dynamics_of_BSD_and_GPL.shtml⟩

Bhattacharya, Jaijit a. *Sourabh Suman*: Analysis of popular open source licenses and their applicability to e-governance; in: Proceedings of the 1st international conference on Theory and practice of electronic governance; New York, NY, USA: ACM, 2007 (= ICEGOV '07) ⟨URL: http://doi.acm.org/10.1145/1328057.1328110⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 978–1–59593–822–0, pp. 254–257

Bianco, Vieri del et al.: The QualiSPo approach to OSS product quality evaluation; [General Chairs: Justin Erenkrantz and Hyrum K. Wright]; In Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 2010, pp. 23–28 ⟨URL: http://doi.acm.org/10.1145/1833272.1833277⟩ – reference download: 2011-12-29, BibWeb/PDF

Bieman, James: Editorial: Free/open source software, silver bullets, and mythical months; in: Software Quality Journal, 14 (2006), pp. 289–290 ⟨URL: http://dx.doi.org/10.1007/s11219-006-0036-3⟩ – reference download: 2012-202-03, BibWeb/PDF

Bird, Christian, *Alex Gourley*, a. *Prem Devanbu*: Detecting Patch Submission and Acceptance in OSS Projects; in: Proceedings of the Fourth International Workshop on Mining Software Repositories; Washington, DC, USA: IEEE Computer Society, 2007 (= MSR '07) ⟨URL: http://dx.doi.org/10.1109/MSR.2007.6⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 0–7695–2950–X, pp. 26:1–26:4

Bird, Christian et al.: Open Borders? Immigration in Open Source Projects; in: Proceedings of the Fourth International Workshop on Mining Software Repositories; Washington, DC, USA: IEEE Computer Society, 2007 (= MSR '07) ⟨URL: http://dx.doi.org/10.1109/MSR.2007.23⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 0–7695–2950–X, pp. 6:1–6:8

Bird, Christian et al.: Latent Social Structure in Open Source Projects; in: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering; New York, NY, USA: ACM, 2008 (= SIGSOFT '08/FSE-16) ⟨URL: http://doi.acm.org/10.1145/1453101.1453107⟩ – reference download: 2012-01-02, BibWeb/PDF, ISBN 978–1–59593–995–1, pp. 24–35

Bitzer, Jürgen, *Wolfram Schrettl*, a. *Philipp J.H. Schröder*: Intrinsic motivation in open source software development; in: Journal of Comparative Economics, 35 (2007), p. 160–169, BibWeb/PDF

Björgvinsson, Tryggvi a. *Helgi Thorbergsson*: Software Development for Governmental Use Utilizing Free and Open Source Software; in: Proceedings of the 1st International Conference on Theory and Practice of Electronic Governance; New York, NY, USA: ACM, 2007 (= ICEGOV '07) ⟨URL: http://doi.acm.org/10.1145/1328057.1328087⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–59593–822–0, pp. 133–140

Black Duck: Top 20 Open Source Licenses; 2014 [n.y], FreeWeb/HTML ⟨URL: http://www.blackducksoftware.com/resources/data/top-20-open-source-licenses⟩ – reference download: 2014-02-11

Böckler, Lina: Mit Freier Software gegen den Wettbewerb; in: *Katharina Vera Boesche, editor:* Variationen im Recht: Beiträge zum Arbeits-, Immaterialgüter-, Infrastruktur-, Lauterkeits-, Unternehmens-, Wettbewerbs- und Zivilrecht; Festbeigabe für Fanz Jürgen Säcker zum 65. Geburstag; Berlin: BWV, 2006, ISBN 3–8305–1234–1, pp. 69 – 76, BibWeb/Copy

Boehm, Barry: A View of 20th and 21st Century Software Engineering; in: Proceedings of the 28th International Conference on Software Engineering; New York, NY, USA: ACM, 2006 (= ICSE '06) ⟨URL: http://doi.acm.org/10.1145/1134285.1134288⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–59593–375–1, pp. 12–29

Boldrin, Michele a. *David K. Levine*: The Case Against Intellectual Property; in: American Economic Review, 92 (2002), No. 2, pp. 209–212

Boldrito, Remo Suppi a. *Josep Jorba Esteve*: GNU/Linux Advanced Administration; coordinated by Josep Jorba Esteve; Barcelona: Free Technology Academy, 2007 ⟨URL: http://www.ftacademy.org/⟩ – reference download: 2012-01-20, FreeWeb/PDF

Boldyreff, Cornelia et al., editors: Open Source Ecosystems: Diverse Communities Interaction; 5th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2009 Skövde, Sweden, June 3-6, 2009; Berlin, Heidelberg and New York: Springer, 2009, BibWeb/PDF, ISBN 978–3–642–02031–5

Bolzern, Mark: A New Project or a GNU Project? in: Linux Journal, 13 May (1995), p. 7:1 ⟨URL: http://dl.acm.org/citation.cfm?id=324822.324829⟩ – reference download: 2011-12-29, BibWeb/HTML

Bonaccorsi, A., M. Merito ans L. Piscitello, a. *C. Rossi*: The 'Open Innovation' Paradigm. How Firms Do Business out of Open Source Software; Copenhagen, 2006, Paper preseneted at the DRUID Summer Conference

Bonaccorsi, A. a. *C. Rossi*: Contributing to the Common Pool Resources in Open Source Software. A Comparison between Individuals and Firms. Pisa, 2003, Sant' Anna School of Advanced Studies; Working Paper

Bonaccorsi, A. a. *C. Rossi*: Why Open Source Software Can Succeed; in: Research Policy, 32 (2003), No. 7, pp. 1243–1258

Bonaccorsi, A. a. *C. Rossi*: Comparing Motivations of Individual Programmers and Firms to Take Part in the Open Source Movement. From Community to Business; Pisa, 2004, Sant' Anna School of Advanced Studies; Working Paper ⟨URL: http://opensource.mit.org/⟩

Bonaccorsi, Andrea et al.: Business Firms' Engagement in Community Projects. Empirical Evidence and Further Developments of the Research; in: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development; Washington, DC, USA: IEEE Computer Society, 2007 (= FLOSS '07) ⟨URL: http://dx.doi.org/10.1109/FLOSS.2007.3⟩, ISBN 0–7695–2961–5, pp. 13–

Booth, David R.: Peer Production and Software. What Mozilla Has To Teach Government; Cambridge (Massachusetts) and London (England): MIT Press, 2010 (= The John D. and Catherine T. MacArthur Foundation Reports on Digital Media and Learning), Print, ISBN 978–0–262–51461–3

Bresson, Jean, *Carlos Agon*, a. *Gérard Assayag*: OpenMusic; Visual Programming Environment for Music Composition, Analysis and Research; in: Proceedings of the 19th ACM International

Conference on Multimedia; New York, NY, USA: ACM, 2011 (= MM '11) ⟨URL: http://doi.acm.org/10.1145/2072298.2072434⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0616–4, pp. 743–746

Bretschneider, Ulrich, *Rainer Glaschick*, a. *Gernot Gräfe*: Ratgeber für die Veröffentlichung von Open-Source-Software durch eine Hochschule; In *Asche et al.*: Open Source. Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen, 2008, pp. 167–188, Print

Brodie, Mark et al.: Support services: persuading employees and customers to do what is in the community's best interest; in: Proceedings of the 2nd international conference on Persuasive technology; Berlin, Heidelberg: Springer-Verlag, 2007 (= PERSUASIVE'07) ⟨URL: http://dl.acm.org/citation.cfm?id=1780402.1780424⟩, ISBN 3–540–77005–4, 978–3–540–77005–3, pp. 121–124

Brown, Peter: EOF: The Free Software Foundation at 20; in: Linux Journal, 137 September (2005), p. 15:1 ⟨URL: http://dl.acm.org/citation.cfm?id=1084783.1084798⟩ – reference download: 2011-12-29, BibWeb/HTML

Brügge, Bernd et al.: Open-Source Software. Eine ökonomische und technische Analyse; Berlin and Heidelberg: Springer, 2004, Print, ISBN 3–540–20366–4

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

Burgess, Guy: Open Source: The Affero General Public License; in: Magazine of the Society for Computers and Law, 19 (2008), No. 4, pp. 42–43, Copy

Bygott, David: David Bygott's Gnu Book. A light-hearted look at the Gnu, or Wildebeest; firstly published 1998; Southerton, Harare: Robert Woollacott, 1992, Print, ISBN 0–7974–1082–1

Bärwolff, M.: Tight Prior Open Source Equilibrium: The Rise of Open Source as a Source of Economic Welfare; in: First Monday, 11 (2006) ⟨URL: http://firstmanday.org/issues/issue11_1/barwolff/index.html⟩

Camp, L. Jean: DRM: Doesn't Really Mean Digital Copyright Management; in: Proceedings of the 9th ACM Conference on Computer and Communications Security; New York, NY, USA: ACM, 2002 (= CCS '02) ⟨URL: http://doi.acm.org/10.1145/586110.586122⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–58113–612–9, pp. 78–87

Campbell-Kelly, Martin: Historical Reflections: Will the Future of Software be Open Source? in: Communications of the ACM, 51 October (2008), No. 10, pp. 21–23 ⟨URL: http://doi.acm.org/10.1145/1400181.1400189⟩ – reference download: 2011-12-29, BibWeb/PDF

Capiluppi, Andrea, *Andres Baravalle*, a. *Nick W. Heap*: From "Community" to "Commercial" FLOSS - the Case of Moodle; [General Chairs: Justin Erenkrantz and Hyrum K. Wright]; In Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 2010, pp. 11–16 ⟨URL: http://doi.acm.org/10.1145/1833272.1833275⟩ – reference download: 2012-02-01, BibWeb/PDF

Capiluppi, Andrea a. *Thomas Knowles*: Software Engineering in Practice: Design and Architecture of FLOSS Systems; 5th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2009 Skövde, Sweden, June 3-6, 2009; In *Boldyreff et al.*: Open Source Ecosystems, 2009, pp. 34–46, BibWeb/PDF

Chawner, Brenda: F/OSS in the Library World: An Exploration; In Proceedings of the Fifth Workshop on Open Source Software Engineering, 2005, pp. 3:1–3:4 ⟨URL: http://doi.acm.org/10.1145/1082983.1083262⟩ – reference download: 2011-12-29, BibWeb/PDF

Cheliotis, Giorgos: From open source to open content: Organization, licensing and decision processes in open cultural production; in: Decision Support Systems, 47 (2009), No. 3, pp. 229–244 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0167923609000578⟩ – reference download: 2012-02-01, BibWeb/PDF

Chen, Shun-ling: Free/Open Source Software. Licensing; Shri Pratap Udyog, Srinivas Puri, New Delhi: Elsevier India, 2006 ⟨URL: http://www.iosn.net/licensing/foss-licensing-primer/foss-licensing-final.pdf⟩ – reference download: 2013-02-02, FreeWeb/PDF, ISBN 978–81–312–0422–1

Chen, Zhixiong a. *Delia Marx*: Experiences with Eclipse IDE in Programming Courses; in: JCSC, 21 December (2005), No. 2, pp. 104–112 ⟨URL: http://dl.acm.org/citation.cfm?id=1089053.1089068⟩ – reference download: 2011-12-29, BibWeb/PDF

Cheung, Gifford et al.: Designing for Discovery: Opening the Hood for Open-Source End User Tinkering; in: Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems; New York, NY, USA: ACM, 2009 (= CHI EA '09) ⟨URL: http://doi.acm.org/10.1145/1520340.1520660⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–60558–247–4, pp. 4321–4326

Chindalia, Sanjanaa: Open source software: The future ahead; in: JOURNAL OF INTELLECTUAL PROPERTY RIGHTS, 13 (2008), pp. 218–224

Chopra, S. a. *S. Dexter*: Free software and the political philosophy of the cyborg world; in: SIGCAS, 37 November (2007), No. 2, pp. 41–52 ⟨URL: http://doi.acm.org/10.1145/1327325.1327328⟩ – reference download: 2011-12-29, BibWeb/PDF

Chopra, S. a. *S. Dexter*: Free software, economic 'realities', and information justice; in: SIGCAS, 39 December (2009), No. 3, pp. 12–26 ⟨URL: http://doi.acm.org/10.1145/1713066.1713067⟩ – reference download: 2011-12-29, BibWeb/PDF

Chopra, Samir a. *Scott Dexter*: The freedoms of software and its ethical uses; in: Ethics and Information Technology, 11 (2009), pp. 287–297 ⟨URL: http://dx.doi.org/10.1007/s10676-009-9191-0⟩, BibWeb/PDF

Christ, Fabian a. *Stefan Sauer*: OSS - Open-Source-Stacks; In *Asche et al.*: Open Source. Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen, 2008, pp. 133–154, Print

Ciffolilli, Andrea: The Economics of Open Source Hijacking and Declining Quality of Digital Information Resources: A Case for Copyleft; 2004 ⟨URL: http://opensource.mit.edu/papers/ciffolili.pdf⟩

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

Colazo, Jorge a. *Yulin Fang*: Impact of License Choice on Open Source Software Development Activity; in: Journal of the American Society for Information Science and Technology, 60 (2009), No. 5, pp. 997–1011, BibWeb/PDF

Coleman, Gabriella: CODE IS SPEECH: Legal Tinkering, Expertise, and Protest among Free and Open Source Software Developers; in: Cultural Anthropology, 24 (2009), No. 3, pp. 420–454 ⟨URL: http://dx.doi.org/10.1111/j.1548-1360.2009.01036.x⟩ – reference download: 2012-02-03, BibWeb/PDF, ISSN 1548–1360

Comino, Stefano a. *Fabio M. Manenti*: Dual licensing in open source software markets; in: Information Economics and Policy, 23 (2011), No. 3–4, pp. 234–242 ⟨URL: http://www.sciencedirect.com/science/article/pii/S016762451100028X⟩ – reference download: 2012-02-01, BibWeb/PDF

Comino, Stefano, *Fabio M. Manenti*, a. *Maria Laura Parisi*: From planning to mature: On the success of open source projects; in: Research Policy, 36 (2007), No. 10, pp. 1575–1586 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0048733307001709⟩ – reference download: 2012-02-01, BibWeb/PDF

copyleft.org: What is copyleft.org; n.l, 2014, FreeWeb/HTML ⟨URL: http://copyleft.org/⟩ – reference download: 2014-12-15

Costa-Soria, Cristóbal a. *Jennifer Pérez*: Teaching Software Architectures and Aspect-Oriented Software Development using Open-Source Projects; in: Proceedings of the 14th Annual ACM

SIGCSE Conference on Innovation and Technology in Computer Science Education; New York, NY, USA: ACM, 2009 (= ITiCSE '09) ⟨URL: http://doi.acm.org/10.1145/1562877.1563027⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–60558–381–5, p. 385

Crowston, Kevin et al.: Effective Work Practices for Software Engineering: Free/Libre Open Source Software Development; in: Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research; New York, NY, USA: ACM, 2004 (= WISER '04) ⟨URL: http://doi.acm.org/10.1145/1029997.1030003⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–58113–988–8, pp. 18–26

Crowston, Kevin a. *James Howison*: The social structure of Free and Open Source software development; in: First Monday 10 (2005), No. 2

Crowston, Kevin et al.: Self-organization of teams for free/libre open source software development; in: Information and Software Technology, 49 (2007), No. 6, pp. 564 – 575 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0950584907000080⟩, ¡ce:title¿Qualitative Software Engineering Research¡/ce:title¿, ISSN 0950–5849

Cuéllar, Luis E.: Open Source License Alternatives for Software Applications; Is it a solution to stop software piracy?; in: Proceedings of the 43rd Annual Southeast Regional Conference; Volume 2, New York, NY, USA: ACM, 2005 ⟨URL: http://doi.acm.org/10.1145/1167253.1167314⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 1–59593–059–0, pp. 269–274

Currion, Paul, *Chamindra de Silva*, a. *Bartel Van de Walle*: Open Source Software For Disaster Management; in: Communications of the ACM, 50 March (2007), No. 3, pp. 61–65 ⟨URL: http://doi.acm.org/10.1145/1226736.1226768⟩ – reference download: 2011-12-29, BibWeb/PDF

Cusumano, Michael A.: Reflections on Free and Open Software; in: Communications of the ACM, 47 October (2004), No. 10, pp. 25–27 ⟨URL: http://doi.acm.org/10.1145/1022594.1022615⟩ – reference download: 2012-202-03, BibWeb/PDF

Dahlander, L.: Appropriation and Appropriability in Open Source Software; in: International Journal of Innovation Management, 9 (2005), No. 3, pp. 259–285

Dahlander, Linus: Penguin in a new suit: a tale of how de novo entrants emerged to harness free and open source software communities; in: Industrial and Corporate Change, 16 (2007), No. 5, pp. 913–943 ⟨URL: http://icc.oxfordjournals.org/content/16/5/913.abstract⟩ – reference download: 2012-202-03, BibWeb/PDF

Dalle, Jean-Michel et al.: Advancing Economic Research on the Free and Open Source Software Mode of Production; In *Wynants a. Cornelius*: How Open is the Future?, 2005, pp. 395–426

David, Paul A. a. *Francesco Rullani*: Dynamics of innovation in an "open source" collaboration environment: lurking, laboring, and launching FLOSS projects on SourceForge; in: Industrial and Corporate Change, 17 (2008), No. 4, pp. 647–710, BibWeb/PDF

Davies, Julius: Measuring Subversions: Security and Legal Risk in Reused Software Artifacts; in: Proceedings of the 33rd International Conference on Software Engineering; New York, NY, USA: ACM, 2011 (= ICSE '11) ⟨URL: http://doi.acm.org/10.1145/1985793.1986025⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 978–1–4503–0445–0, pp. 1149–1151

Davis, Donald a. *Iffat Jabeen*: Learning in the GNU/Linux Community; in: Proceedings of the 2011 Conference on Information Technology Education; New York, NY, USA: ACM, 2011 (= SIGITE '11) ⟨URL: http://doi.acm.org/10.1145/2047594.2047600⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–1017–8, pp. 21–26

Davis, Mike et al.: Linux and Open Source in the Academic Enterprise; in: Proceedings of the 28th annual ACM SIGUCCS conference on User services: Building the future; New York, NY, USA: ACM, 2000 (= SIGUCCS '00) ⟨URL: http://doi.acm.org/10.1145/354908.354923⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 1–58113–229–8, pp. 65–69

de Laat, Paul B.: Copyright or copyleft? An analysis of property regimes for software development; in: Research Policy, 34 (2005), pp. 1511–1532, BibWeb/PDF

De Nicolò, Christopher: Open Source Software - Rechtliche Aspekte nach deutschem und italienischem Recht. Eine rechtsvergleichende Studie. Dissertation; Regensburg: Universität Regensburg, 2010, Print

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

Deike, Thies: Open Source Software: IPR-Fragen und Einordnung ins deutsche Rechtssystem; in: CR [Computer und Recht], (2003), pp. 9ff

Deitcher, Avi: The challenges of open source in the enterprise; in: Linux Journal, 195 July (2010), pp. Article No. 3 ⟨URL: http://dl.acm.org/citation.cfm?id=1883478.1883481⟩ – reference download: 2011-12-28, BibWeb/HTML

Dempsey, Bert J. et al.: A quatitative profile of a community of Open Source Linux developers; North Carolina: University of North Carolina at Chapel Hill, School of Information and Librabry Science, 1999 (= (= [University of North Carolina] Technical Report TR 1999-05))

Dempsey, Bert J. et al.: Who Is an Open Source Software Developer? in: Communications of the ACM, 45 (2002), No. 2, pp. 67–72 ⟨URL: http://doi.acm.org/10.1145/503124.503125⟩ – reference download: 2011-12-28, BibWeb/PDF

Deodhar, Swanand J., *K. B. C. Saxena*, a. *Mikko Ruohonen*: Firm-Oriented Success Factors of an Open Source Software (OSS) Product; [General Chairs: Justin Erenkrantz and Hyrum K. Wright]; In Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 2010, pp. 1–4 ⟨URL: http://doi.acm.org/10.1145/1833272.1833273⟩ – reference download: 2011-12-29, BibWeb/PDF

Determann, Lothar: Softwarekombinationen unter der GPL; in: GRUR Int. (slg: Gewerblicher Rechtsschutz und Urheberrecht, Internationaler Teil, 2006), (2006), pp. 645 – 653

Di Penta, Massimiliano et al.: An Exploratory Study of the Evolution of Software Licensing; in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering; Volume 1, New York, NY, USA: ACM, 2010 ⟨URL: http://doi.acm.org/10.1145/1806799.1806824⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 978–1–60558–719–6, pp. 145–154

DiBona, Chris, *Cooper C*, a. *D. Stone, editors*: Open Sources 2.0: The Continuing Evolution; Sebastopol CA: O'Reilly, 2005

DiBona, Chris, *Sam Ockman*, a. *Mark Stone, editors*: Open Sources. Voices from the Open Source Revolution; Beijing u.a.: O'Reilly, 1999

Diedrich, Oliver: Die Geschichte von Linux; 2011, FreeWeb/PDF ⟨URL: http://www.heise.de/open/artikel/Die-Geschichte-von-Linux-1329997.html⟩ – reference download: 20110826

Dionisio, John David N. et al.: An Open Source Software Culture in the Undergraduate Computer Science Curriculum; in: SIGCSE Bulletin, 39 June (2007), No. 2, pp. 70–74 ⟨URL: http://doi.acm.org/10.1145/1272848.1272888⟩ – reference download: 2011-12-29, BibWeb/PDF

Djordjevic, Valle et al., editors: Urheberrecht im Alltag. Kopieren, bearbeiten, selber machen; Bonn: Bundeszentrale für politische Bildung, 2008, Print, ISBN 978–3–89331–812–4

Dobb, Dr.: It All About The License; in: Informationweek, n.V. (2009), No. 1253, p. 46, Copy

Doernhoefer, Mark: Surfing the Net for Software Enginerring Notes; in: SIGSOFT Software Engineering Notes, 35 (2010), No. 4, pp. 8–16, BibWeb/PDF

Donnelly, Francis P.: Evaluating open source GIS for libraries; in: Library Hi Tech, 28 (2010), No. 1, pp. 131–151 ⟨URL: http://www.emeraldinsight.com/0737-8831.htm⟩ – reference download: 2012-02-13, BibWeb/PDF

Donorfio, Brian: The Politics of "Free": Open Source Software in Government; in: JCSC, 19 (2004), No. 5, pp. 279–280 ⟨URL: http://dl.acm.org/citation.cfm?id=1060081.1060117⟩ – reference download: 2011-12-29, BibWeb/PDF

Dorman, David: The Case for Open Source Software in the Library Market; in: Ubiquity, January (2004), p. 4:1 ⟨URL: http://doi.acm.org/10.1145/985600.985601⟩ – reference download: 2011-12-29, BibWeb/HTML

Dougherty, William C. a. *Audrey Schadt*: Linux Is for Everyone; Librarians Included! in: The Journal of Academic Librarianship, 36 (2010), No. 2, pp. 173–175 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0099133310000108⟩ – reference download: 2012-02-09, BibWeb/PDF

Douglas, David: A bundle of software rights and duties; in: Ethics and Information Technology, 13 (2011), pp. 185–197 ⟨URL: http://dx.doi.org/10.1007/s10676-010-9229-3⟩ – reference download: 2012.02.09, BibWeb/PDF

Drossou, Olga, *Stefan Krempl*, a. *Andreas Poltermann*: Der Kampf um die Innovationsfreiheit: Der Big Bang des Wissens und seine Sprengkraft. Plädoyer für einen offenen Umgang mit Wissen im Interesse der Innovationskraft von Wirtschaft und Gesellschaft; Editorial; In *Drossou, Krempl, a. Poltmann*: Die wunderbare Wissensvermehrung, 2006, pp. 1–10, Print

Drossou, Olga, *Stefan Krempl*, a. *Andreas Poltmann, editors*: Die wunderbare Wissensvermehrung. Wie Open Innovation unsere Welt revolutioniert; (= Telepolis) Hannover: Heise, 2006, Print, ISBN 3–936931–38–0

Eckl, Julian: Die politische Ökonomie der Wissenschaftsgesellschaft. Geistige Eigentumsrechte und die Frage des Zugangs zu Ideen; Marburg: Tectum Verlag, 2004, Print, ISBN 3–8288–8735–X

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

Eclipse Foundation: CPL to EPL Conversion; 2013 [n.y. of the page itself], FreeWeb/HTML ⟨URL: http://www.eclipse.org/legal/cpl2epl/⟩ – reference download: 2013-02-20

Economides, Nicholas a. *Evangelos Katsamakas*: Two-Sided Competition of Proprietary vs. Open Source Technology Platforms and the Implications for the Software Industry; in: Management Science, 52 (2006), No. 7, pp. 1057–1071

Elkemann-Reusch, Ilva: Die erzwungene Gegengabe; in: ZGE / IPJ, 2 (2010), pp. 413–452, Copy

Elliott, Margaret, *Mark S. Ackerman*, a. *Walt Scacchi*: Knowledge Work Artifacts: Kernel Cousins for Free/Open Source Software Development; in: Proceedings of the 2007 international ACM conference on Supporting group work; New York, NY, USA: ACM, 2007 (= GROUP '07) ⟨URL: http://doi.acm.org/10.1145/1316624.1316650⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–59593–845–9, pp. 177–186

Elliott, Margaret S. a. *Walt Scacchi*: Free Software Developers as an Occupational Community: Resolving Conflicts and Fostering Collaboration; in: Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work; New York, NY, USA: ACM, 2003 (= GROUP '03) ⟨URL: http://doi.acm.org/10.1145/958160.958164⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–58113–693–5, pp. 21–30

Ellis, Jason a. *Jean-Paul Van Belle*: Open Source Software Adoption by South African MSEs: Barriers and Enablers; in: Proceedings of the 2009 Annual Conference of the Southern African Computer Lecturers' Association; New York, NY, USA: ACM, 2009 (= SACLA '09) ⟨URL: http://doi.acm.org/10.1145/1562741.1562746⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–60558–683–0, pp. 41–49

Engelfriet, Arnoud: Tools of the Trade[:] Choosing an Open Source License; in: IEEE Software, 27 (2010), No. 1, pp. 48–49, Copy

Epplin, J.: Using GPL Software in Embedded Applications; ⟨URL: http://www.linux.devices.com/articles/AT916119242.html⟩

Ernst, Stefan: Die Verfügbarkeit des Sourcecodes; in: MultiMedia und Recht, 4 (2001), pp. 208–213

Euler, Ellen: Creative Commons: Mehr Innovation durch die Öffnung des Urheberechts? In *Drossou, Krempl, a. Poltmann*: Die wunderbare Wissensvermehrung, 2006, pp. 147–158, Print

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

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

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

Evans, David S. a. *Bernard J. Reddy*: Government Preferences for Promoting Open-Source Software: A Solution in Search of a Problem; in: 9 Mich Telecomm. Tech. L. Rev. 313 (2003), pp. 313–394

Ezeala, Adanna, *Hyunju Kim*, a. *Loretta A. Moore*: Open Source Software Development: Expectations and Experience from a Small Development Project; in: Proceedings of the 46th Annual Southeast Regional Conference on XX; New York, NY, USA: ACM, 2008 (= ACM-SE 46) ⟨URL: http://doi.acm.org/10.1145/1593105.1593168⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–60558–105–7, pp. 243–246

Fantl, Stephen: Copyleft or Copyright. Into the new paradigm; in: MacTech Magazine, 16 (2000), No. 10, pp. 98–100, Copy

Fehr, Ernst a. *Simon Gächter*: Fairness and Retaliation: The Economics of Reciprocity; in: Journal of Economic Perspectives, 14 (2000), pp. 159–181

Feig, Michael: Einführung in GNU; München and Wien: Carl Hanser Verlag, 1996 (= Unix easy), Print, ISBN 3–446–18311–6

Feller, Joseph: Meeting challenges and surviving success: the 2nd workshop on open source software engineering; evaluate the complete workshop results; in: Proceedings of the 24th International Conference on Software Engineering; New York, NY, USA: ACM, 2002 (= ICSE '02) ⟨URL: http://doi.acm.org/10.1145/581339.581436⟩, ISBN 1–58113–472–X, pp. 669–670

Feller, Joseph a. *Brian Fitzgerald*: A Framework Analysis of the Open Source Software Development Paradigm; in: Proceedings of the twenty first international conference on Information systems; Atlanta, GA, USA: Association for Information Systems, 2000 (= ICIS '00) ⟨URL: http://dl.acm.org/citation.cfm?id=359640.359723⟩, BibWeb/PDF, pp. 58–69

Feller, Joseph et al.: Collaboration, Conflict and Control: The 4th Workshop on Open Source Software Engineering; evaluate worksup results; in: Proceedings of the 26th International Conference on Software Engineering; Washington, DC, USA: IEEE Computer Society, 2004 (= ICSE '04) ⟨URL: http://dl.acm.org/citation.cfm?id=998675.999508⟩, ISBN 0–7695–2163–0, pp. 764–765

Feller, Joseph et al.: Collaboration, conflict and control: report on the 4th workshop on open source software engineering; in: SIGSOFT Softw. Eng. Notes, 30 May (2005), pp. 1–2 ⟨URL: http://doi.acm.org/10.1145/1061874.1061885⟩, ISSN 0163–5948

Feller, Joseph et al.: Taking stock of the bazaar: the third workshop on open source software engineering; in: SIGSOFT Softw. Eng. Notes, 28 November (2003), pp. 5–5 ⟨URL: http://doi.acm.org/10.1145/966221.966227⟩, ISSN 0163–5948

Feller, Joseph, *Brian Fitzgerald*, a. *André van der Hoek*: 1st workshop on open source software engineering; evaluate workshop results; in: Proceedings of the 23rd International Conference on Software Engineering; Washington, DC, USA: IEEE Computer Society, 2001 (= ICSE '01) ⟨URL: http://dl.acm.org/citation.cfm?id=381473.381660⟩, ISBN 0–7695–1050–7, pp. 780–781

Feller, Joseph, *Brian Fitzgerald*, a. *André van der Hoek*: Making sense of the bazaar: 1st workshop on open source software engineering; in: SIGSOFT Softw. Eng. Notes, 26 November (2001), pp. 51–52 ⟨URL: http://doi.acm.org/10.1145/505532.505543⟩, ISSN 0163–5948

Feller, Joseph et al.: Open source application spaces: the 5th workshop on open source software engineering; evaluatre workshop results; in: Proceedings of the 27th international conference on Software engineering; New York, NY, USA: ACM, 2005 (= ICSE '05) ⟨URL: http://doi.acm.org/10.1145/1062455.1062619⟩, ISBN 1–58113–963–2, pp. 694–694

Feller, Jospeph a. *Brian Fitzgerald*: Understanding Open Source Software Development; Reading, Mass., London?: Addison-Wesley, 2002

Fielding, Roy T.: Shared leadership in the Apache Project; in: Communications of the ACM, 42 (1999), No. 4, pp. 42–43

Fielding, Roy T.: Software Architecture in an Open Source World; in: Proceedings of the 27th International Conference on Software Engineering; New York, NY, USA: ACM, 2005 (= ICSE '05) ⟨URL: http://doi.acm.org/10.1145/1062455.1062474⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–58113–963–2, pp. 43–43

Fink, Martin: The business and economics of Linux and open source; Upper Saddle River, N.J.: Prentice Hal, 2003

Fitzgerald, Brian: The Transformation of Open Source Software; in: MIS Quarterly, 30 (2006), No. 3, pp. 587–598

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

Fosfuri, Andrea, *Marco S. Giarratana*, a. *Alessandra Luzzi*: The Penguin Has Entered the Building: The Commercialization of Open Source Software Products; in: OrganizationScience, 19 March-April (2008), No. 2, p. 292–305, BibWeb/PDF

Fox, Laurie a. *Shawn Plummer*: Opening the Lines of Communications with Open Source Software; in: Proceedings of the 34th Annual ACM SIGUCCS Fall Conference; New York, NY, USA: ACM, 2006 (= SIGUCCS '06) ⟨URL: http://doi.acm.org/10.1145/1181216.1181242⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–59593–438–3, pp. 114–117

Franck, Egon a. *Carola Jungwirth*: Die Governance von Open Source Projekten; in: Zeitschrift für Betriebswirtschaft, 73 (2003), No. 5 (Ergänzungsheft), pp. 1 –21

Franky, María Consuelo: Agile Management and Development of Software Projects based on Collaborative Environments; in: SIGSOFT Software Engineering Notes, 36 May (2011), No. 3, pp. 1–6 ⟨URL: http://doi.acm.org/10.1145/1968587.1968605⟩ – reference download: 2011-12-29, BibWeb/PDF

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

Free Software Foundation: GNU Library General Public License [version 2.0]; 1991 [n.y. of the html page itself], FreeWeb/HTML ⟨URL: http://www.gnu.org/licenses/old-licenses/lgpl-2.0.html⟩ – reference download: 2013-03-25

Free Software Foundation: GNU Lesser General Public License [Version 2.1]; 1999 [n.y. of the html page itself], FreeWeb/HTML ⟨URL: http://www.gnu.org/licenses/lgpl-2.1.html⟩ – reference download: 2013-03-06

Free Software Foundation: GNU General Public License [version 3]; 2007 [n.y. of the html

page itself], FreeWeb/HTML ⟨URL: http://www.gnu.org/licenses/gpl.html⟩ – reference download: 2013-03-06

Free Software Foundation: GNU Lesser General Public License [version 3]; 2007 [n.y. of the html page itself], FreeWeb/HTML ⟨URL: http://www.gnu.org/copyleft/lesser.html⟩ – reference download: 2013-03-06

Free Software Foundation: GNU Operating System[:] Licenses; 2011, FreeWeb/HTML ⟨URL: http://www.gnu.org/licenses/⟩ – reference download: 2013-03-25

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

Free Software Foundation: What is free software? The Free Software Definition; 2015 [n.y.], FreeWeb/HTML ⟨URL: https://www.gnu.org/philosophy/free-sw.en.html⟩ – reference download: 2015-02-20

Friedman, Batya et al.: Development of a Privacy Addendum for Open Source Licenses: Value Sensitive Design in Industry; in: *Paul Dourish* a. *Adrian Friday, editors:* UbiComp 2006: Ubiquitous Computing; [Proceedings of the] 8th International Conference, UbiComp 2006 Orange County, CA, USA, September 17-21, 2006; Berlin, Heidelberg, a. New York: Springer, 2006 (= Lecture Notes in Computer Science, [Vol./No.] 4206), BibWeb/PDF, ISBN 978–3–540–39634–5, pp. 194–211

Fujita, Kunihiko a. *Yasuyuki Tsukada*: An Analysis of Interoperability between Licenses; in: Proceedings of the Tenth Annual ACM Workshop on Digital Rights Management; New York, NY, USA: ACM, 2010 (= DRM '10) ⟨URL: http://doi.acm.org/10.1145/1866870.1866884⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0091–9, pp. 61–72

Funk, Axel a. *Georg Zeitfang*: Die GNU General Public License, Version3; in: CR [Computer und Recht], (2007), pp. 617 – 624

Funk, Axel a. *Gregor Zeitfang*: Die GNU General Public License, Version 3; in: Computer und Recht, 23 (2007), No. 10, pp. 617–624, BibWeb/Copy

Fuse Source Team, The: How to Use Open Source Integration Software Safely in the Enterprise. Analysis of potential risks and how to protect your IT environment; October 2010, FreeWeb/PDF [File received by a promotion campaign of itwhitepapers.com. The article refers to http:// www. fusesource. com/. But we didn't retrieve the paper there.] ⟨URL: http://www.itwhitepapers.com/?option=com_categoryreport\&task=viewabstract\&pathway=no\&autodn=1\&title=14770\&crv=0\&src=5053\&ctg=410\&cmp=3732\&yld=0\&pi=1628115⟩ – reference download: 2011-08-24

Gallini, Nancy a. *Suzanne Scotchmer*: Intellectual Property: When Is It the Best Incentive System? in: NBER Innovation Policy & the Economy, 2 (2002), No. 1, pp. 51–77

Gambardella, Alfonso a. *Bronwyn H. Hall*: Proprietary versus public domain licensing of software and research products; in: RP, 35 (2006), No. 6, pp. 875–892 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0048733306000643⟩ – reference download: 2012-02-09, BibWeb/PDF

Gandal, Neil a. *Chaim Fershtman*: Open Source Projectsd: Output per Contributor and Restrictive Licensing; o.O.: ???, 2004 (= (= CEPR Working Paper 2650))

Gassmann, Oliver a. *Martin A. Bader*: Patentmanagement: Innovationen erfolgreich nutzen und schützen; Berlin: ???, 2005

Geese, Elmar: Innovation und freie Software; In *Drossou, Krempl, a. Poltmann*: Die wunderbare Wissensvermehrung, 2006, pp. 77–84, Print

Gehring, Robert A. a. *Bernd Lutterbeck, editors*: Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell; Berlin: Lehmanns Media, 2004 ⟨URL: http:

//www.opensourcejahrbuch.de/download/jb2004/OpenSourceJahrbuch2004.pdf⟩ – reference download: 2011-08-29, Print & FreeWeb/PDF, ISBN 3–936427–78–X

Gerber, Aurona, *Onkgopotse Molefe*, a. *Alta van der Merwe*: Documenting Open Source Migration Processes for Re-use; in: Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists; New York, NY, USA: ACM, 2010 (= SAICSIT '10) ⟨URL: http://doi.acm.org/10.1145/1899503.1899512⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 978–1–60558–950–3, pp. 75–85

Gerlach, Carsten: Praxisprobleme der Open-Source-Lizenzierung; in: CR [Computer und Recht], (2006), pp. 649 – 654

German, Daniel M.: Using software distributions to understand the relationship among free and open source software projects; in: Proceedings of the Fourth International Workshop on Mining Software Repositories; Washington, DC, USA: IEEE Computer Society, 2007 (= MSR '07) ⟨URL: http://dl.acm.org/citation.cfm?id=1268983.1269038⟩, BibWeb/PDF, ISBN 0–7695–2950–X, p. 24

German, Daniel M. a. *Jesús M. González-Barahona*: An Empirical Study of the Reuse of Software Licensed under the GNU General Public License; conference contribution; In *Boldyreff et al.*: Open Source Ecosystems, 2009, pp. 185–198, BibWeb/PDF

German, Daniel M. a. *Ahmed E. Hassan*: License Integration Patterns: Addressing License Mismatches in Component-Based Development; in: Proceedings of the 31st International Conference on Software Engineering; Washington, DC, USA: IEEE Computer Society, 2009 (= ICSE '09) ⟨URL: http://dx.doi.org/10.1109/ICSE.2009.5070520⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 978–1–4244–3453–4, pp. 188–198

German, Daniel M., *Yuki Manabe*, a. *Katsuro Inoue*: A Sentence-Matching Method for Automatic License Identification of Source Code Files; in: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering; New York, NY, USA: ACM, 2010 (= ASE '10) ⟨URL: http://doi.acm.org/10.1145/1858996.1859088⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0116–9, pp. 437–446

German, Daniel M., *Jens H. Webber*, a. *Massimiliano Di Penta*: Lawful Software Engineering; in: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research; New York, NY, USA: ACM, 2010 (= FoSER '10) ⟨URL: http://doi.acm.org/10.1145/1882362.1882390⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0427–6, pp. 129–132

Gilroy, Bernard Michael a. *Tobias Volpert*: Die Funktionen eines Patentsystems und ihre Bedeutung für Unternehmensausgründungen aus Hochschulen; In *Asche et al.*: Open Source. Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen, 2008, pp. 21–39, Print

Gobeille, Robert: The FOSSology Project; in: Proceedings of the 2008 international working conference on Mining software repositories; New York, NY, USA: ACM, 2008 (= MSR '08) ⟨URL: http://doi.acm.org/10.1145/1370750.1370763⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 978–1–60558–024–1, pp. 47–50

Godfrey, Michael a. *Qiang Tu*: Growth, Evolution, and Structural Change in Open Source Software; in: Proceedings of the 4th International Workshop on Principles of Software Evolution; New York, NY, USA: ACM, 2001 (= IWPSE '01) ⟨URL: http://doi.acm.org/10.1145/602461.602482⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 1–58113–508–4, pp. 103–106

Goeminne, Mathieu a. *Tom Mens*: A Framework for Analysing and Visualising Open Source Software Ecosystems; in: Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE); New York, NY, USA: ACM, 2010 (= IWPSE-EVOL '10) ⟨URL: http://doi.acm.org/10.1145/

1862372.1862384⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0128–2, pp. 42–47

Goldmann, R. a. *R.P. Gabriel*: Innovation Happens Elsewhere. Opne Source as Business Strategy; San Francisco: Elsevier, 2005

Gomulkiewicz, Robert W.: De-Bugging Open Source Software Licensing; in: University of Pittsburgh Law Review, 64 (2002), pp. 75–99, BibWeb/PDF

González-Barahona, Jesús M., *Joaquín Seoane Pascual*, a. *Gregorio Robles*: Introduction to Free Software; coordinated by Jordi Mas Hernández and David Megías Jiménez; Oberta de Catalunya: Free Technology Academy, 2009 ⟨URL: http://www.ftacademy.org/materials/fsm/1#1⟩ – reference download: 2012-01-20, FreeWeb/PDF

Goode, Sigi: Something for nothing: management rejection of open source software in Australia's top firms; in: Information & Management, 669–681 (2005), p. 42, BibWeb/PDF

Gordon, Thomas F.: Analyzing Open Source License Compatibility Issues with Carneades; in: Proceedings of the 13th International Conference on Artificial Intelligence and Law; New York, NY, USA: ACM, 2011 (= ICAIL '11) ⟨URL: http://doi.acm.org/10.1145/2018358.2018364⟩ – reference download: 2011-12-29, BIbWeb/PDF, ISBN 978–1–4503–0755–0, pp. 51–55

Grassmuck, Volker: Open Source - Betriebssysteme für eine freiheitliche Gesellschaft; ⟨URL: http://www.waste.informatik.hu-berlin.de/Grassmuck/Texts/OSS-Tutzing-5-00.html⟩

Grassmuck, Volker: Freie Software. Geschichte, Dynamiken und gesellschaftliche Bezüge; Berlin, 2000

Grassmuck, Volker: Lizenzmodelle; 2000 ⟨URL: http://www.mikro.org/Events/OS/text/lizenzen.html⟩

Grassmuck, Volker: Freie Software. Zwischen Privat- und Gemeineigentum; Themen und Materialien; Bonn: Bundeszentrale für politische Bildung, 2002, Print, ISBN 3–89331–432–6

Green, Collin et al.: Leveraging Open-Source Software in the Design and Development Process; in: Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems; New York, NY, USA: ACM, 2009 (= CHI EA '09) ⟨URL: http://doi.acm.org/10.1145/1520340.1520433⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 978–1–60558–247–4, pp. 3061–3074

Green, Eric Lee: Economics of Open Source Software; 1998 ⟨URL: http://badtux.org/home/eric/editorial/economics.php⟩

Green, Lisa a. *Heather Meeker*: Open Software Licenses: Part II; in: Intellectual Property Strategist 10 (1999)

Greve, Georg C. F.: Geschichte und Philosophie des GNU Projekts; ⟨URL: http://www.gnu.org/philosophy/greeve-clown.html⟩

Grodzinsky, F. S. a. *M. C. Bottis*: Private Use as Fair Use: Is it Fair? in: SIGCAS, 37 November (2007), No. 2, pp. 11–24 ⟨URL: http://doi.acm.org/10.1145/1327325.1327326⟩ – reference download: 2011-12-29, BibWeb/PDF

Gräfe, Gernot: Open-Source-Software und Open-Source-Portale - Potentiale für die Softwareentwicklung in Hochschulen und den Ergebnistransfer in die Praxis; In *Asche et al.*: Open Source. Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen, 2008, pp. 55–72, Print

Grützmacher, Malte: Open-Source-Software - die GNU General Public License / Lizenzbestimmungen im Umfeld des neuen Schuld- und Urhebervertragsrechts; in: ITRB (IT-Rechtsberater), (2002), pp. 84ff

Grützmacher, Malte: Open Source Software - BSD Copyright and Apache Software. Copyright statt Copyleft; in: ITRB, o.A. (2006), No. 5, pp. 1008–112, Copy

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

Gull, Daniel: Valuation of Discount Options in Software License Agreements; in: BISE, 4 (2011), pp. 221–230 ⟨URL: http://dx.doi.org/10.1007/s12599-011-0170-8⟩ – reference download: 2012-02-09, BibWeb/PDF

Gurbani, Vijay K., *Anita Garvert*, a. *James D. Herbsleb*: Managing a Corporate Open Source Software Asset; in: Commununications of the ACM, 53 February (2010), No. 2, pp. 155–159 ⟨URL: http://doi.acm.org/10.1145/1646353.1646392⟩ – reference download: 2011-12-29, BibWeb/PDF

Gutsche, Jörg: Ökonomische Analyse offener Software; Mannheim: Universität Mannheim, 2006, BibWeb/PDF

Gutwin, Carl, *Reagan Penner*, a. *Kevin Schneider*: Group Awareness in Distributed Software Development; in: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work; New York, NY, USA: ACM, 2004 (= CSCW '04) ⟨URL: http://doi.acm.org/10.1145/1031607.1031621⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–58113–810–5, pp. 72–81

Haase, H.: Die Patentierbarkeit von Computersoftware; Hamburg, 2003

Haddad, I.: Adopting an Open Source Approach to Software Development, Distribution, and Licensing; in: Enterprise Open Source Magazine, (2007) ⟨URL: http://opensource.sys-con.com/read/318776.htm⟩

Haddad, Ibrahim: Open-Source Compliance; in: Linux Journal, 185 September (2009), p. 5:1 ⟨URL: http://dl.acm.org/citation.cfm?id=1610564.1610569⟩ – reference download: 2011-12-29, BibWeb/HTML

Hahn, Robert, editor: Government Policy toward Open Source Software; AEI-Brooking Joint Centre (Org.) 2002

Hamerly, Jim, *Tom Paquin*, a. *Susan Walton*: Freeing the Source: The Story of Mozilla; In *DiBona, Ockman, a. Stone*: Open Sources, 1999, pp. 197–206

Hammouda, Imed et al.: Open source legality patterns: architectural design decisions motivated by legal concerns; in: Proceedings of the 14th International Academic MindTrek Conference: Envisioning Future Media Environments; New York, NY, USA: ACM, 2010 (= MindTrek '10) ⟨URL: http://doi.acm.org/10.1145/1930488.1930533⟩ – reference download: 2012-01-06, BibWeb/PDF, ISBN 978–1–4503–0011–7, pp. 207–214

Hardaway, Donald E.: Sharing Research in the 21st Century: Borrowing a Page from Open Source Software; in: Communications of the ACM, 48 August (2005), No. 8, pp. 125–128 ⟨URL: http://doi.acm.org/10.1145/1076211.1076216⟩ – reference download: 2011-12-29, BibWeb/PDF

Hars, Alexander a. *Shaosong Ou*: Working for free? Motivations for participating in open source projects; in: International Journal of Electronic Commerce, 6 (2002), No. 3, pp. 25–39

Hauge, Oyvind et al.: An Empirical Study on Selection of Open Source Software - Preliminary results; In Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009, pp. 42–47 ⟨URL: http://dx.doi.org/10.1109/FLOSS.2009.5071359⟩ – reference download: 2011-12-29, BibWeb/PDF

Hauge, Oyvind a. *Sven Ziemer*: Providing Commercial Open Source Software: Lessons Learned; conference contribution; In *Boldyreff et al.*: Open Source Ecosystems, 2009, pp. 70–82, BibWeb/PDF

Hawkins, R. E.: The Economics of Open Source Software in a Competitive Firm: Why Give It Away For Free? in: Netnomics, 6 (2004), pp. 103–117

Heap, Nicholas: OSI-Referenzmodell ohne Geheimnis; translated by G & U, Flensburg; Hannover: Heise, 1994, Print, ISBN 3–88229–045–5

Hecker, F.: Setting up Shop: The Business of Open Source Software; in: IEEE Software, Jan/Feb (1999), pp. 46–61

Heffan, Ira V.: Copyleft: Licensing Collaborative Works in the Digital Age; in: Stanford Law Review, 1997 (49), pp. 1487–1521 ⟨URL: http://www.jstor.org/stable/1229351⟩ – reference download: 2012-02-09, BibWeb/PDF

Heise online: Deutsches Gericht bestätigt Wirksamkeit der GPL; 2004 ⟨URL: http://www.heise,de/newsticker/meldung/49377⟩

Heise online: Der Streit um Softwarepateht; 2007 ⟨URL: http://www.heise,de/ct/hintergrund/meldung/61230⟩

Hemel, Armijn et al.: Finding Software License Violations Through Binary Code Clone Detection; in: Proceedings of the 8th Working Conference on Mining Software Repositories; New York, NY, USA: ACM, 2011 (= MSR '11) ⟨URL: http://doi.acm.org/10.1145/1985441.1985453⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0574–7, pp. 63–72

Henkel, Joachim: Software development in embedded Linux; In *Uhr, Esswein, a. Schoop*: Wirtschaftsinformatik 2003 / Band II, 2003, pp. 81–99

Henkel, Joachim: The Jukebox Mode of Innovation; A Model of Commercial Open Source Development; 2004 ⟨URL: http://opensource.mit.edu/papers/henkel.pdf⟩

Henkel, Joachim: Open source software from commercial firms; Tools, complements, and collective invention; in: Zeitschrift für Betriebswirtschaft, 4/2004 (2004), No. 4 Ergänzungsheft, pp. 1–23

Henkel, Joachim: Patterns of Free Revealing; Balancing Code Sharing and Protection in Commercial Open Source Development; 2004 ⟨URL: http://opensource.mit.edu/papers/henkel2.pdf⟩

Henkel, Joachim: Offene Innovationsprozesse. Die kommerzielle Entwicklung von Open-Source-Software; Wiesbaden: Deutscher Universitäts-Verlag, 2007 (= Gabler Edition Wissenschaft), BibWeb/PDF, ISBN 978–3–8350–0978–3

Herraiz, Israel et al.: The Processes of Joining in Global Distributed Software Projects; in: Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner; New York, NY, USA: ACM, 2006 (= GSD '06) ⟨URL: http://doi.acm.org/10.1145/1138506.1138513⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–59593–404–9, pp. 27–33

Herraiz, Israel, *Gregorio Robles*, a. *Jesus M. Gonzalez-Barahona*: Towards Predictor Models for large Libre Software Projects; in: Proceedings of the 2005 Workshop on Predictor Models in Software Engineering; New York, NY, USA: ACM, 2005 (= PROMISE '05) ⟨URL: http://doi.acm.org/10.1145/1082983.1083168⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN –159593–125–2, pp. 1–6

Herraiz, Israel, *Gregorio Robles*, a. *Jesus M. Gonzalez-Barahona*: Research Friendly Software Repositories; in: Proceedings of the Joint International and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) Workshops; New York, NY, USA: ACM, 2009 (= IWPSE-Evol '09) ⟨URL: http://doi.acm.org/10.1145/1595808.1595814⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–60558–678–6, pp. 19–24

Hertel, Guido, *Sven Niedner*, a. *Stefanie Herrmann*: Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel; in: Research Policy, 32 (2003), pp. 1159–1177

Hill, Benjamin Mako: Samir Chopra, Scott D. Dexter, Decoding Liberation: The Promise of Free and Open Source Software; in: Minds Mach. 18 June (2008), pp. 297–299 ⟨URL: http://dl.acm.org/citation.cfm?id=1375424.1375428⟩, ISSN 0924–6495

Hislop, Gregory W. et al.: Using Open Source sSoftware to Engage Students in Computer Science Education; in: Proceedings of the 40th ACM technical symposium on Computer science education; New York, NY, USA: ACM, 2009 (= SIGCSE '09) ⟨URL: http://doi.acm.org/10.1145/1508865.1508915⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–60558–183–5, pp. 134–135

Hissam, Scott A. et al., editors: Open Source Systems: Grounding Research; 7th IFIP WG 2.13 International Conference, OSS 2011; (= IFIP Advances in Information and Communication Technology, [Vol./No.] 365) Heidelberg, Dordrecht, London u. NewYork: Springer, 2011, BibWeb/PDF, ISBN 978–3–642–24418–6

Hoeren, Thomas: Anmerkungen zum Urteil vom 19.5.2004 des LG München I zur Wirksamkeit einer GPL-Lizenz; in: CR [Computer und Recht], (2004), pp. 776–778

Hoeren, Thomas: Internetrecht; April 2011, Web/PDF? ⟨URL: http://www.uni-muenster.de/Jura.itm/hoeren/lehre/materialien⟩

Hofmann, Susanne, *Sven Pfeiffer*, a. *Urs Walter*: Open Source School. Neue Synergien zwischen Schule und Kiez in Gropiusstadt. Architektur als sozialer Katalysator; Berlin, 2010, BibWeb/PDF ⟨URL: http://opus.kobv.de/tuberlin/volltexte/2010/2841/pdf/9783798322738_content.pdf⟩ – reference download: 2011-07-30

Horne, Natasha: Open Source Software Licensing: Using Copyright Law to Encourage Free Use; in: Georgia State University Law Review, (2001), pp. 863–891

Horns, A.: Der Patentschutz für software bezogene Erfindungen im Verhätnis zur 'Open Source'-Software; in: Zeitschrift für Rechtsinformatik, (2000) ⟨URL: http://www.jurpc.de/aufsatz/2000223.htm⟩

Howland, John E.: Software Freedom, Open Software and the Undergraduate Computer Science Curriculum; in: JCSC, 15 March (2000), No. 3, pp. 293–301 ⟨URL: http://dl.acm.org/citation.cfm?id=1852563.1852604⟩ – reference download: 2011-12-29, BibWeb/PDF

Howland, John E.: Managing Computer Science Laboratories Using Open Software; in: JCSC, 16 March (2001), No. 3, pp. 117–126 ⟨URL: http://dl.acm.org/citation.cfm?id=374685.374726⟩ – reference download: 2011-12-29, BibWeb/PDF

Hubbard, Jordan: Open Source to the Core; in: Queue, 2 (2004), pp. 24–31 ⟨URL: http://doi.acm.org/10.1145/1005062.1005064⟩, BibWeb/PDF

ifross: FAQ; 2011, FreeWeb/HTML ⟨URL: http://www.ifross.org/faq-haeufig-gestellte-fragen⟩ – reference download: 2011-09-05

ifross: Ziele, Aufgaben, Geschichte; 2011, FreeWeb/HTML ⟨URL: http://www.ifross.org/node/16⟩ – reference download: 2011-09-05

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

ifross et al.: Die GPL kommentiert und erklärt; Beijing, Cambridge, Farnham [etc ..]: O'Reilly, 2005, Print, ISBN 3–89721–389–3

Ilardi, Terry J.: Common OSS License Problems; n.l, 2010, FreeWeb/PDF ⟨URL: http://www2.aipla.org/html/spring/2010/papers/Ilardi_Paper.pdf⟩ – reference download: 2014-12-16

Imhorst, Christian: Die Anarchie der Hacker. Richard Stallman und die Freie-Software-Bewegung; Marburg: Tectum Verlag, 2004, Print, ISBN 3–8288–8769–4

Izurieta, Clemente a. *James Bieman*: The Evolution of FreeBSD and Linux; in: Proceedings of the 2006 ACM/IEEE international Symposium on Empirical Software Engineering; New York, NY, USA: ACM, 2006 (= ISESE '06) ⟨URL: http://doi.acm.org/10.1145/1159733.1159765⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 1–59593–218–6, pp. 204–211

Jackson, Darla W.: Thinking about Technology . . . Watson, Answer Me This: Will You Make Librarians Obsolete or Can I Use Free and Open Source Software and Cloud Computing

to Ensure a Bright Future? in: Law Library Journal, 103 (2011), No. 3, pp. 497–504, BibWeb/PDF

Jacobs, Stephen, *Clif Kussmaul*, a. *Mihaela Sabin*: Free and Open Source Software in Computing Education; in: Proceedings of the 2011 Conference on Information Technology Education; New York, NY, USA: ACM, 2011 (= SIGITE '11) ⟨URL: http://doi.acm.org/10.1145/2047594.2047606⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–1017–8, pp. 41–42

Jaeger, Till: Copyright oder Copyleft; in: Computerwoche Spezial, 27 (2000), No. 4, p. 36 ⟨URL: http://www.ifross,de/ifross_html/art6.html⟩

Jaeger, Till: GPL und Haftung: Ohne Verantwortung? in: Linux-Magazin, (2000), No. 5, pp. 134ff

Jaeger, Till: Die GPL kommentiert und erklärt; hrsg; v. ifross; Köln, 2005

Jaeger, Till a. *Axel Metzger*: Open Source Software und deutsches Urheberrecht; in: GRUR Int. (1999), pp. 839ff

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

Jaeger, Till a. *Axel Metzger*: Open Content-Lizenzen nach deutschem Recht; in: MultiMedia und Recht, (2003), pp. 431ff

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

Jaeger, Till a. *Axel Metzger*: Die neue Version 3 der GNU general Public License; in: GRUR (Gerwerblicher Rechtsschutz und Urheberrecht), (2008), pp. 130–137

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

Jaeger, Till a. *Carsten Schulz*: Gutachten zu ausgewählten rechtlichen Aspekten der Open Source Software - im Rahmen des Projektes 'NOW - Nutzung des Open Source-Konzepts in Wirtschaft und Industrie'; Feb 2005

Janamanchi, Balaji et al.: The State and Profile of Open Source Software Projects in health and medical informatics; in: International Journal of Medical Informatics, 78 (2009), No. 7, pp. 457–472 ⟨URL: http://www.sciencedirect.com/science/article/pii/S1386505609000318⟩ – reference download: 2012-02-09, BibWeb/PDF

Jansson, Kurt, *Patrick Danowski*, a. *Jakob Voss*: Wikipedia: Kreative Anarchie für den freien Informations- und Wissensaustausch; In *Drossou, Krempl, a. Poltmann*: Die wunderbare Wissensvermehrung, 2006, pp. 159–167, Print

Jendroska, Dirk: Arbeitsgestaltung in der Softwareentwicklung: Ein empirischer Vergleich subjektiver Arbeitsmerkmale in proprietären und Open Source Softwareprojekten; Dissertation; Münster: Philosophischen Fakultät der Westfälischen Wilhelms Universität zu Münster, 2010, BibWeb/PDF

Johnson, Justin P.: Open Source Software: Private Provision of a Public Good; in: Journal of Economics & Management Strategy, 11 (2002), No. 4, pp. 637–663

Johnson, Michael K.: Licenses and Copyright; in: Linux Journal, 29 September (1996), p. 3:1 ⟨URL: http://dl.acm.org/citation.cfm?id=326350.326353⟩ – reference download: 2011-12-28, BibWeb/HTML

Johnson-Eilola, Johndan: Open Source Basics: Definitions, Models, and Questions; in: Proceedings of the 20th Annual International Conference on Computer Documentation; New York, NY, USA: ACM, 2002 (= SIGDOC '02) ⟨URL: http://doi.acm.org/10.1145/584955.584967⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–58113–543–2, pp. 79–83

Johri, Aditya, *Oded Nov*, a. *Raktim Mitra*: "Cool" or "Monster"? Company Takeovers and Their Effect on Open Source Community Participation; in: Proceedings of the 2011 iConference;

New York, NY, USA: ACM, 2011 (= iConference '11) ⟨URL: http://doi.acm.org/10.1145/1940761.1940806⟩, ISBN 978–1–4503–0121–3, pp. 327–331

Jones, Paul: Open (Source)ing the Doors for Contributor-run Digital Libraries; in: Communications of the ACM, 44 May (2001), No. 1, pp. 45–46 ⟨URL: http://doi.acm.org/10.1145/374308.374337⟩ – reference download: 2011-12-49, BibWeb/PDF

Karels, Michael J.: Commercializing Open Source Software; in: Queue, 1 July/August (2003), pp. 46–55 ⟨URL: http://doi.acm.org/10.1145/945074.945125⟩ – reference download: 2011-12-28, BibWeb/PDF

Karus, Siim a. *Harald Gall*: A Study of Language Usage Evolution in Open Source Software; in: Proceedings of the 8th Working Conference on Mining Software Repositories; New York, NY, USA: ACM, 2011 (= MSR '11) ⟨URL: http://doi.acm.org/10.1145/1985441.1985447⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 978–1–4503–0574–7, pp. 13–22

Kelty, Christopher M: Free Software/Free Science; in: First Monday, 6 (2001), No. 12, p. o.A.

Kelty, Christopher M.: Two Bits: The cultural Significance of Free Software; ???, 2008

Kennedy, D. M.: A primer on open source licensing legal issues: copyright, copyleft and copyfuture; 2001 ⟨URL: http://www.denniskennedy.com/opensourcedmk.pdf⟩

Kern, W. a. *F. Rammig*: Eine Einführung zum Open Source Konzept aus Sicht der wirtschaftlichen und rechtlichen Aspekte; Paderborn; in: C-LAB Report, 2 (2003), vielleicht Buch?

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

Keuffel, Warren: License Overload; in: Software Development, 14 (2006), No. 2, p. 56, Copy

Keßler, Steffen a. *Paul Alpar*: Customization of Open Source Software in Companies; 5th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2009 Skövde, Sweden, June 3-6, 2009; In *Boldyreff et al.*: Open Source Ecosystems, 2009, pp. 129–142, BibWeb/PDF

Kienle, Holger M. et al.: Intellectual Property Aspects of Web Publishing; in: Proceedings of the 22nd annual international conference on Design of communication: The engineering of quality documentation; New York, NY, USA: ACM, 2004 (= SIGDOC '04) ⟨URL: http://doi.acm.org/10.1145/1026533.1026569⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 1–58113–809–1, pp. 136–144

Kilamo, Terhi: The Community Game: Learning Open Source Development Through Participatory Exercise; in: Proceedings of the 14th International Academic MindTrek Conference: Envisioning Future Media Environments; New York, NY, USA: ACM, 2010 (= MindTrek '10) ⟨URL: http://doi.acm.org/10.1145/1930488.1930500⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0011–7, pp. 55–60

Kirschner, Bryan: Building a Balanced Scorecard for Open Source Policy and Strategy: A Case Study of the Microsoft Experience; in: Proceedings of the 2nd International Conference on Theory and Practice of Electronic Governance; New York, NY, USA: ACM, 2008 (= ICEGOV '08) ⟨URL: http://doi.acm.org/10.1145/1509096.1509142⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–60558–386–0, pp. 226–231

Kitcat, Jason: Source Availability and E-voting: An Advocate Recants; in: Communications of the ACM, 47 October (2004), No. 10, pp. 65–67 ⟨URL: http://doi.acm.org/10.1145/1022594.1022625⟩ – reference download: 2011-12-29, BibWeb/PDF

Klemm, Martin: GPL Version 3.0; in: Innovation und internationale Rechtspraxis, o.A. (2009), pp. 363–381, Copy

Koch, Frank A.: Urheber- und kartellrechtliche Aspekte der Nutzung von Open-Source-Software (I); in: CR [Computer und Recht], (2000), pp. 273ff

Koch, Frank A.: Urheber- und kartellrechtliche Aspekte der Nutzung von Open-Source-Software (II); in: CR [Computer und Recht], (2000), p. 333

Koch, Frank A.: Probleme beim Wechsel zur neuen Version 3 der General Public License (Teil 1); in: ITRB (IT-Rechtsberater), (2007), pp. 261–263

Koch, Frank A.: Probleme beim Wechsel zur neuen Version 3 der General Public License (Teil 2); in: ITRB (IT-Rechtsberater), (2007), pp. 285–288

Koeglin, Olaf: Entfesseltes Wissen - Creative Commons und der Versuch, das GPL-Prinzip für jede Schöpfung anzuwenden; in: Linux-Magazin, (2003), No. 10, p. 70

Koeglin, Olaf: Die Nutzung von Open Source Software unter neuen GPL Versionen nach der 'any later Version'-Klausel; in: CR [Computer und Recht], (2008), pp. 137–143

Koeglin, Olaf a. *Axel Metzger*: Urheber- und Lizenzrecht im Bereich von Open-Source-Software; in Open Source Jahrbuch 2004; 2004, pp. 293ff

Koenig, J.: Seven Open Source Business Strategies for mCompetitive Advantage; in: IT Manager's Journal, (2004) ⟨URL: http://management.itmanagersjournal.com/article?sid=04/05/10/2052216⟩

Koglin, Olaf: Opensourcerecht. Die urheber- und schuldrechtlichen Beziehungen zwischen Lizenzgeber und Lizenznehmer bei Open Source Software am Beispiel der General Public License (GPL); Frankfurt am Main: Peter Lang, 2007 (= Schriften zum Wirtschafts- und Medienrecht, Steuerrecht und Zivilprozeßrecht, [Vol./No.] 31), Print, ISBN 978–3–631–56308–3

Kogut, B. a. *A. Metiu*: Open Source Software Development and Distributed Innovation; in: Oxford Review of Economic Policy, 17 (2001), pp. 248–264

Koponen, Timo a. *Virpi Hotti*: Open Source Software Maintenance Process Framework; In Proceedings of the Fifth Workshop on Open Source Software Engineering, 2005, pp. 4:1–4:5 ⟨URL: http://doi.acm.org/10.1145/1082983.1083265⟩ – reference download: 2011-12-28, BibWeb/PDF

Kreutzer, Till: Anmerkungen zum Urteil vom 19.5.2004 des LG München I zur Wirksamkeit einer GPL-Lizenz; in: MultiMedia und Recht, (2004), pp. 695–698

Kreutzer, Till: Software und Spiele kopieren[:] Das Lizenzmodell entscheidet; In *Djordjevic et al.*: Urheberrecht im Alltag, 2008, pp. 29–33, Print

Kreutzer, Till: Software veröffentlichen[:] Wem gehören die Rechte? In *Djordjevic et al.*: Urheberrecht im Alltag, 2008, pp. 163–167, Print

Kreutzer, Till: Softwarelizenzen - Beispiele[:] Und welche Lizenz nehme ich jetzt? In *Djordjevic et al.*: Urheberrecht im Alltag, 2008, pp. 176–179, Print

Krishnamurthya, Sandeep a. *Arvind K. Tripathi*: Monetary donations to an open source software platform; in: Research Policy, 38 (2009), pp. 404–414

Krogstie, Birgit R.: Power Through Brokering: Open Source Community Participation in Software Engineering Student Projects; in: Proceedings of the 30th International Conference on Software Engineering; New York, NY, USA: ACM, 2008 (= ICSE '08) ⟨URL: http://doi.acm.org/10.1145/1368088.1368201⟩ – reference download: 2011-12-29, BIbWeb/PDF, ISBN 978–1–60558–079–1, pp. 791–800

Kugler, Petra: Coordinating Innovation: Evidence from Open Source Development; Dissertation; St. Gallen: University of. St. Gallen, 2005, Print

Kuhlen, Rainer: Open Innovation: Teil einer nachhaltigen Wissensökonomie; In *Drossou, Krempl, a. Poltmann*: Die wunderbare Wissensvermehrung, 2006, pp. 12–23, Print

Kuhn, Bradley M. et al.: Copyleft and the GNU General Public License: A Comprehensive Tutorial and Guide; n.l, 2014, FreeWeb/PDF ⟨URL: http://copyleft.org/guide/comprehensive-gpl-guide.pdf⟩ – reference download: 2014-12-15

Kumar, Vineet, *Brett R. Gordon*, a. *Kannan Srinivasan*: Competitive Strategy for Open Source Software; in: MARKETING SCIENCE, 30 (2011), No. 6, pp. 1066–1078, BibWeb/PDF

Käs, Simone: Rethinking industry practice. The emergence of openness in the embedded component industry; München: Pro BUSINESS, 2008, Print, ISBN 978–3–86805–256–5

Lacy, Sarah: Open Source: Now It's an Ecosystem; 2005 ⟨URL: http://www.businessweek.com/technology/content/oct2005/tc2005103_0519_tc_218.htm⟩

Lakhani, Karim R. a. *Eric Hippel*: How Open Source software works: "Free" user-to-user assistance; 2002, MIT Sloan School of Management Working Paper

Lakhani, Karim R. a. *Robert G. Wolf*: Why Hackers Do What They Do: Understanding Motivation Effort in Free/Open Source Software Projects; (= MIT Sloan School of Management Working, Paper 4425-03), 2003

Laroque, Chrstoph, *Andre Döring*, a. *Thorsten Timm*: 'Give or Let Buy': Kritische Überlegungen eines Software-Ingeneurs zur Veröffentlichung von Software als Open-Source-Projekte; In *Asche et al.*: Open Source. Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen, 2008, pp. 155–166, Print

Lavazza, Luigi et al.: Predicting OSS Trustworthiness on the Basis of Elementary Code Sssessment; in: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement; New York, NY, USA: ACM, 2010 (= ESEM '10) ⟨URL: http://doi.acm.org/10.1145/1852786.1852834⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0039–1, pp. 36:1–36:4

Lawrie, Tony a. *Cristina Gacek*: Issues of Dependability in Open Source Software Development; in: SIGSOFT Software Engineering Notes, 27 May (2002), No. 3, pp. 34–37 ⟨URL: http://doi.acm.org/10.1145/638574.638584⟩

Lee, Samuel, *Nina Moisa*, a. *Marco Weiss. An Economic Analysis*: Open Source as a Signalling Device; Frankfurt a.M.: Goethe-University Frankfur/Main, 2003 (= Working Paper Series: Finance and Accounting, [Vol./No.] 102), BibWeb/PDF

Lee, Samuel, *Nina Moisa*, a. *Marco Weiss*: Conditions for Open Source as a Signalling Device; Frankfurt a.M.: Goethe-University Frankfur/Main, 2004 (= Working Paper Series: Finance and Accounting), BibWeb/PDF

Lejeune, Mathias: Rechtsprobleme bei der Lizenzierung von Open Source Software nach der GNU GPL; in: ITRB (IT-Rechtsberater), 1 (2003), pp. 10–12

Lelli, Francesco a. *Mehdi Jazayeri*: Community Support for Software Development in Small Groups: the Initial Steps; in: Proceedings of the 2nd international workshop on Social software engineering and applications; New York, NY, USA: ACM, 2009 (= SoSEA '09) ⟨URL: http://doi.acm.org/10.1145/1595836.1595840⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–60558–682–3, pp. 15–22

Lemley, Mark A. a. *Ziv Shafir*: Who Chooses Open-Source Software? in: University of Chicago Law Review, 78 (2011), pp. 139–163, BibWeb/PDF

Lenarcic, John a. *Eric C. Mousset*: The Open Source Singularity: A Postmodernist View; in: Selected papers from conference on Computers and philosophy - Volume 37; Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2003 (= CRPIT '03) ⟨URL: http://dl.acm.org/citation.cfm?id=1082145.1082157⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 1–920–68219–8, pp. 73–77

Lerner, Josh a. *Jean Tirole*: The open source movement: Key research questions; in: European Economic Review, 45 (2001), pp. 819–826, BibWeb/PDF

Lerner, Josh a. *Jean Tirole*: The Scope of Open Source Licensing; in: JLEO, 21 (2005), No. 1, pp. 20–56, BibWeb/PDF

Lerner, Joshua a. *Jean Tirole*: Some simple economics of Open Source; in: Journal of Industrial Economics, 50 (2002), No. 2, pp. 197–234

Levy, S.: Hackers; USA: Penguin, 2001

Li, Yan, *Chuan Hoo Tan*, a. *Hock Hai Teo*: Firm-Specificity and Organizational Learning-related Scale on Investment in Internal Human Capital for Open Source Software Adoption; in: Proceedings of the 2008 ACM SIGMIS CPR Conference on Computer Personnel Doctoral

Consortium and Research; New York, NY, USA: ACM, 2008 (= SIGMIS CPR '08) ⟨URL: http://doi.acm.org/10.1145/1355238.1355244⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 978–1–60558–069–2, pp. 22–29

Li, Yan et al.: Motivating Open Source Software Developers: Influence of Transformational and Transactional Leaderships; in: Proceedings of the 2006 ACM SIGMIS CPR Conference on Computer Personnel Research: Forty Four Years of Computer Personnel Research: Achievements, Challenges & the Future; New York, NY, USA: ACM, 2006 (= SIGMIS CPR '06) ⟨URL: http://doi.acm.org/10.1145/1125170.1125182⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–59593–349–2, pp. 34–43

Li, Yan et al.: Open Source Software Adoption: Motivations of Adopters and Amotivations of Non-adopters; in: SIGMIS Database, 42 May (2011), No. 2, pp. 76–94 ⟨URL: http://doi.acm.org/10.1145/1989098.1989103⟩ – reference download: 2011-12-29, BibWeb/PDF

Li, Yung-Ming, *Jhih-Hua Jhang-Li*, a. *Yen-Chun Liu*: Optimal Strategies of IT Consulting Firms: The Impact of License Fee and Open Source; in: Proceedings of the 10th International Conference on Electronic Commerce; New York, NY, USA: ACM, 2008 (= ICEC '08) ⟨URL: http://doi.acm.org/10.1145/1409540.1409594⟩ – reference download: 2011-12-29, ISBN 978–1–60558–075–3, pp. 40:1–40:7

Lin, Yi-Hsuan et al.: Open Source Licenses and the Creative Commons Framework: License Selection and Comparison; in: JISE, 22 (2006), pp. 1–17, BibWeb/PDF

Lin, Yu-Wei a. *Enrico Zini*: Free/libre open source software implementation in schools: Evidence from the field and implications for the future; in: Computers & Education, 50 (2008), No. 3, pp. 1092–1102 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0360131506001722⟩, BibWeb/PDF, ISSN 0360–1315

Lindman, J., *M. Rossi*, a. *A. Puustell*: Matching Open Source Software Licenses with Corresponding Business Models; in: Software, IEEE, 28 july-aug. (2011), No. 4, pp. 31 –35, ISSN 0740–7459

Lindman, Juho, *Juha-Pekka Juutilainen*, a. *Matti Rossi*: Beyond the Business Model: Incentives for Organizations to Publish Software Source Code; conference contribution; In *Boldyreff et al.*: Open Source Ecosystems, 2009, pp. 47–56, BibWeb/PDF

Lovett, Jayne: Open Source - A Practical Solution; in: Proceedings of the 35th Annual ACM SIGUCCS Fall Conference; New York, NY, USA: ACM, 2007 (= SIGUCCS '07) ⟨URL: http://doi.acm.org/10.1145/1294046.1294099⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–59593–634–9, pp. 221–223

Lundell, Björn, *Brian Lings*, a. *Edvin Lindqvist*: Open source in Swedish companies: where are we? in: Information Systems Journal, 20 (2010), p. 519–535, BibWeb/PDF

Lutterbeck, Bernd, *Matthias Baerwolff*, a. *Robert A. Gehring, editors*: Open Source Jahrbuch 2006. Zwischen Softwareentwicklung und Gesellschaftsmodell; Berlin: Lehmanns Media, 2006 ⟨URL: http://www.opensourcejahrbuch.de/download/jb2006/OpenSourceJahrbuch2006_online.pdf⟩ – reference download: 2011-10-17, Print & FreeWeb/PDF, ISBN 3–86541–135–5

Lutterbeck, Bernd, *Matthias Baerwolff*, a. *Robert A. Gehring, editors*: Open Source Jahrbuch 2007. Zwischen Softwareentwicklung und Gesellschaftsmodell; Berlin: Lehmanns Media, 2007 ⟨URL: http://www.opensourcejahrbuch.de/download/jb2007/OpenSourceJahrbuch2007_online.pdf⟩ – reference download: 2011-10-17, Print & FreeWeb/PDF, ISBN 978–3–86541–191–4

Lutterbeck, Bernd, *Matthias Baerwolff*, a. *Robert A. Gehring, editors*: Open Source Jahrbuch 2008. Zwischen Softwareentwicklung und Gesellschaftsmodell; Berlin: Lehmanns Media, 2008 ⟨URL: http://www.opensourcejahrbuch.de/download/jb2008/osjb08.pdf⟩ – reference download: 2011-10-17, Print & FreeWeb/PDF, ISBN 978–3–86541–271–3

Lutterbeck, Bernd a. *Robert A. Gehring Matthias Bärwolff , editors*: Open Source Jahrbuch; Berlin: Lehmanns Media, 2005 ⟨URL: http://...mitaufnehmen..⟩

Maaß, C a. *E. Schern*: Softwarepatente; in: Das Wirtschaftsstudium, 33 (2004), No. 10, pp. 1026–1028

Maaß, C. a. *E. Schern*: Software-Lizenzierung; in: Das Wirtschaftsstudium, 34 (2005), No. 2, pp. 185–188

Maaß, Christian: Zur Bedeutung des Urheber- und Patanterechts in der quelloffenen Softwareentwicklung; In *Asche et al.*: Open Source. Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen, 2008, pp. 41–54, Print

MacCormack, A., *Risnack J*, a. *C. Y. Baldwin*: Exploring the Structure of Complex Software Design: An Empricial Study of Open Source and Proprietary Code; in: Management Science, 52 (2006), pp. 1015–1030

Mahler, Marcus: Open Source Software: The Success of an Alterntaive Intellectual Property Incentive Paradigm; in: Fordham Intellectual Property, Media & Entertainmeint Law Journal, 21 (2000), pp. 619–646

Maldonado, Edgar: The Process of Introducing FLOSS in the Public Administration: The Case of Venezuela; in: JAIS, 11 (2010), No. 11, pp. 756–783, BibWeb/PDF

Manabe, Yuki, *Yasuhiro Hayase*, a. *Katuro Inoue*: Evolutional Analysis of Licenses in FOSS; in: Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE); New York, NY, USA: ACM, 2010 (= IWPSE-EVOL '10) ⟨URL: http://doi.acm.org/10.1145/1862372.1862391⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 978–1–4503–0128–2, pp. 83–87

Mancinelli, Fabio et al.: Managing the Complexity of Large Free and Open Source Package-Based Software Distributions; in: Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering; Washington, DC, USA: IEEE Computer Society, 2006 ⟨URL: http://dl.acm.org/citation.cfm?id=1169218.1169319⟩, ISBN 0–7695–2579–2, pp. 199–208

Mann, Florian et al.: Open Access Publishing In Science; in: Communications of the ACM, 52 March (2009), pp. 135–139 ⟨URL: http://doi.acm.org/10.1145/1467247.1467279⟩

Mannaert, Herwig a. *Kris Ven*: The Use of Open Source Software Platforms by Independent Software Vendors: Issues and Opportunities; In Proceedings of the Fifth Workshop on Open Source Software Engineering, 2005, pp. 7:1–7:4 ⟨URL: http://doi.acm.org/10.1145/1082983.1083266⟩ – reference download: 2011-12.29, BibWeb/PDF

Marly, Jochen: Praxishandbuch Softwarerecht; 5th edition. München: Beck, 2009

Marmorstein, Robert: Open Source Contribution As An Effective Software Engineering Class Project; in: Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education; New York, NY, USA: ACM, 2011 (= ITiCSE '11) ⟨URL: http://doi.acm.org/10.1145/1999747.1999823⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0697–3, pp. 268–272

Martins Melo, Felipe a. *Pereira, Jr.*: A Component-Based Open-Source Framework for General-Purpose Recommender Systems; in: Proceedings of the 14th international ACM Sigsoft symposium on Component based software engineering; New York, NY, USA: ACM, 2011 (= CBSE '11) ⟨URL: http://doi.acm.org/10.1145/2000229.2000239⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 978–1–4503–0723–9, pp. 67–72

McAllister, Neil: Licence to Profit [. Hybrid Open Source Licensing]; in: New Architect and Web Techniques (www.newarchitectmag.com), 8 (2003), p. np., Copy

McGowan, D.: Legal Implications of Open Source Software; in: University Illinois Law Review, (2001), pp. 241–304

McGowan, David: The Tory Anarchism of F/OSS Licensing; in: University of Chicago Law Review, 78 (2011), pp. 207–223, BibWeb/PDF

McInerney, Paul-Brian: Technology Movements and the Politics of Free/Open Source Software; in: Science, Technology & Human Values, 34 (2009), No. 2, pp. 206–233 ⟨URL: http://sth.sagepub.com/content/34/2/206.abstract⟩, BibWeb/PDF

Megias, David et al.: Free Technology Academy: a European initiative for distance education about Free Software and Open Standards; in: Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education; New York, NY, USA: ACM, 2009 (= ITiCSE '09) ⟨URL: http://doi.acm.org/10.1145/1562877.1562904⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 978–1–60558–381–5, pp. 70–74

Meretz, Stefan: Linux & Co : freie Software - Ideen für eine andere Gesellschaft; Neu-Ulm: ???, 2000

Metzger, Axel: Frei ab 18 Jahre; in: Linux-Magazin, (2000), No. 11, pp. 52ff

Metzger, Axel: Anmerkungen zum Urteil vom 19.5.2004 des LG München I zur Wirksamkeit einer GPL-Lizenz; in: CR [Computer und Recht], (2004), pp. 778–780

Michaelson, Jay: There's no such thing as a Free (Software) Lunch; in: Queue, 2 May (2004), pp. 40–47 ⟨URL: http://doi.acm.org/10.1145/1005062.1005066⟩ – reference download: 2011-12-29, BibWeb/PDF

Microsoft: Einige Fragen zur GNU General Public License (GPL), die sich jedes Unternehmen stellen sollte; in: ??? (2001)

Mitre: Use of Free and Open Source Software (FOSS) in the U.S. Department of Defense; ⟨URL: http://www.egovos.org/pdf/dodfoss.pdf⟩

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

Mockus, Audris, *Roy T. Fielding*, a. *James Herbsleb*: A Case Study of Open Source Software Development: the Apache Server; in: Proceedings of the 22nd international conference on Software engineering; New York, NY, USA: ACM, 2000 (= ICSE '00) ⟨URL: http://doi.acm.org/10.1145/337180.337209⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–58113–206–9, pp. 263–272

Mockus, Audris, *Roy T. Fielding*, a. *James D. Herbsleb*: Two Case Studies of Open Source Software Development: Apache and Mozilla; in: Transactions on Software Engineering Methodology, 11 July (2002), No. 3, pp. 309–346 ⟨URL: http://doi.acm.org/10.1145/567793.567795⟩

Moglen, Even a. *Mishi Choudhary*: Software Freedom Law Center Guide to GPL Compliance, 2nd Edition; 2014, FreeWeb/HTML ⟨URL: https://www.softwarefreedom.org/resources/2014/SFLC-Guide_to_GPL_Compliance_2d_ed.html⟩ – reference download: 2014-12-15

Moglen, Peter: Anarchism triumphant: Free Software and the Death of Copyright; in: First Monday, 48 (1999), p. o.A.

Monden, A. et al.: Guilty or Not Guilty: Using Clone Metrics to Determine Open Source Licensing Violations; in: Software, IEEE, 28 march-april (2011), No. 2, pp. 42 –47, ISSN 0740–7459

Montante, Robert: A Survey of Portable Software; in: JCSC, 24 January (2009), No. 3, pp. 19–24 ⟨URL: http://dl.acm.org/citation.cfm?id=1409873.1409879⟩ – reference download: 2011-12-29, BibWeb/PDF

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

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

Moody, Glyn: Interview with Eric Raymond; in: Linux Journal, 165 January (2008), p. 5:1 ⟨URL: http://dl.acm.org/citation.cfm?id=1344189.1344194⟩ – reference download: 2011-12-29, BibWeb/HTML

Morasca, Sandro, *Davide Taibi*, a. *Davide Tosi*: Towards Certifying the Testing Process of Open-Source Software: New Challenges or Old Methodologies? In Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009, pp. 25–30 ⟨URL: http://dx.doi.org/10.1109/FLOSS.2009.5071356⟩ – reference download: 2011-12-29, BibWeb/PDF

Morasca, Sandro, *Davide Taibi*, a. *Davide Tosi*: Towards certifying the testing process of Open-Source Software: New challenges or old methodologies? in: Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development; Washington, DC, USA: IEEE Computer Society, 2009 (= FLOSS '09) ⟨URL: http://dx.doi.org/10.1109/FLOSS.2009.5071356⟩, ISBN 978–1–4244–3720–7, pp. 25–30

Morelli, Ralph a. *Trishan de Lanerolle*: Foss 101: Engaging Introductory Students in the Open Source Movement; in: Proceedings of the 40th ACM technical symposium on Computer science education; New York, NY, USA: ACM, 2009 (= SIGCSE '09) ⟨URL: http://doi.acm.org/10.1145/1508865.1508977⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 978–1–60558–183–5, pp. 311–315

Morgan, Lorraine a. *Patrick Finnegan*: Open Innovation in Secondary Software Firms: An Exploration of Managers' Perceptions of open Source Software; in: SIGMIS Database, 41 February (2010), No. 1, pp. 76–95 ⟨URL: http://doi.acm.org/10.1145/1719051.1719056⟩, BibWeb/PDF

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

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

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

Mtsweni, Jabu a. *Elmarie Biermann*: An investigation into the implementation of open source software within the SA government: an emerging expansion model; in: Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology; New York, NY, USA: ACM, 2008 (= SAICSIT '08) ⟨URL: http://doi.acm.org/10.1145/1456659.1456677⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 978–1–60558–286–3, pp. 148–158

Müller Molina, Arnoldo José a. *Takeshi Shinohara*: On Approximate Matching of Programs for Protecting Libre Software; in: Proceedings of the 2006 Conference of the Center for Advanced Studies on Collaborative research; New York, NY, USA: ACM, 2006 (= CASCON '06) ⟨URL: http://doi.acm.org/10.1145/1188966.1188994⟩ – reference download: 2011-12-29, BibWeb/PDF, pp. 1–14

Mundhenke, Jens: Wettbewerbswirkungen von Open-Source-Software und offenen Standards auf Softwaremärkten; Berlin, Heidelberg, and New York: Springer, 2007 (= Kiel Studies, [Vol./No.] 338), Print, ISBN 978–540–71415–6

Munga, Neeshal, *Thomas Fogwill*, a. *Quentin Williams*: The Adoption of Open Source Software in Business Models: A Red Hat and IBM Case Study; in: Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists; New York, NY, USA: ACM, 2009 (= SAICSIT '09) ⟨URL: http://doi.acm.org/10.1145/1632149.1632165⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 978–1–60558–643–4, pp. 112–121

Mustaquim, Moyen Mohammad: A Systems Thinking Model for Open Source Software Development in Social Media; in: Proceedings of the International Workshop on Modeling Social Media; New York, NY, USA: ACM, 2010 (= MSM '10) ⟨URL: http://doi.acm.org/10.1145/1835980.1835987⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0229–6, pp. 7:1–7:2

Mustonen, Mikko: Copyleft - the economics of Linux and other open source software; in: Information Economics and Policy, 15 (2003), No. 1, pp. 99–121 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0167624502000902⟩ – reference download: 2012-02-09, BibWeb/PDF & Copy

Mustonen, Mikko: Essays on the Economics of Information and Communication Technologies: Copyleft, Networks and Compatibility; Ph. D thesis, University of Helsinki, Department of Economics, Faculty of Social Sciences, 2003

Mustonen, Mikko: Why do firms support the development of substitute copyleft programs? Volume 15 of *Mustonen: Copyleft - the economics of Linux and other open source software, 2003*⟨URL: http://www.sciencedirect.com/science/article/pii/S0167624502000902⟩ – reference download: 2012-02-09, pp. ??–??, BibWeb/PDF & Copy

Mustonen, Mikko: When Does a Firm Support Substitute Open Source Programming? in: Journal of Economics & Management Strategy, 14 (2005), No. 1, pp. 121–138

Müller, Martin: Open Source - kurz & gut; Köln, 1999 ⟨URL: http://www.oreilly.de/german/freebooks/os_tb/toc.html⟩

Müller-Seitz, Gordon a. *Guido Reger*: Is open source software living up to its promises? Insights for open innovation management from two open source software-inspired projects; in: R&D Management, 39 (2009), No. 4, pp. 372–381 ⟨URL: http://dx.doi.org/10.1111/j.1467-9310.2009.00565.x⟩ – reference download: 2012-02-09, BibWeb/PDF

Nadah, Nadia, *Mélanie Dulong de Rosnay*, a. *Bruno Bachimont*: Licensing Digital Content With A Generic Ontology: Escaping From The Jungle of Rights Expression Languages; in: Proceedings of the 11th International Conference on Artificial Intelligence and Law; New York, NY, USA: ACM, 2007 (= ICAIL '07) ⟨URL: http://doi.acm.org/10.1145/1276318.1276330⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–59593–680–6, pp. 65–69

Nagy, Del, *Areej M. Yassin*, a. *Anol Bhattacherjee*: Organizational adoption of open source software: barriers and remedies; in: Communications of the ACM, 53 (2010), pp. 148–151 ⟨URL: http://doi.acm.org/10.1145/1666420.1666457⟩, BibWeb/PDF

Nakakoji, Kumiyo et al.: Evolution Patterns of Open-Source Software Systems and Communities; in: Proceedings of the International Workshop on Principles of Software Evolution; New York, NY, USA: ACM, 2002 (= IWPSE '02) ⟨URL: http://doi.acm.org/10.1145/512035.512055⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–58113–545–9, pp. 76–85

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

Neumann, Peter G.: Inside Risks: Robust Open-Source Software; in: Communications of the ACM, 42 February (1999), No. 2, p. 128 ⟨URL: http://doi.acm.org/10.1145/293411.293491⟩, BibWeb/PDF

Nilendu, P. a. *T. R. Madanmohan*: Competing on Open Source: Strategies and Practise; 2002 ⟨URL: http://opensource.mit.edu/papers/madanmohan.pdf⟩

Noll, John a. *Wei-Ming Liu*: Requirements Elicitation in Open Source Software Development: A Case Study; [General Chairs: Justin Erenkrantz and Hyrum K. Wright]; In Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 2010, pp. 35–40 ⟨URL: http://doi.acm.org/10.1145/1833272.1833279⟩ – reference download: 2011-12-28, BibWeb/PDF

Nordquist, Pete, *Anna Petersen*, a. *Angelina Todorova*: License Tracing in Free, Open, and Proprietary Software; in: JCSC, 19 December (2003), No. 2, pp. 101–112 ⟨URL: http://dl.acm.org/citation.cfm?id=948785.948802⟩ – reference download: 2011-12-28, BibWeb/PDF

Nov, Oded a. *George Kuk*: Open source content contributors' response to free-riding: The effect of personality and context; in: Computers in Human Behavior, 24 (2008), pp. 2848–2861, BibWeb/PDF

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

Oezbek, Christopher, *Lutz Prechelt*, a. *Florian Thiel*: The Onion has Cancer: Some Social Network Analysis Visualizations of Open Source Project Communication; [General Chairs: Justin Erenkrantz and Hyrum K. Wright]; In Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 2010, pp. 5–10 ⟨URL: http://doi.acm.org/10.1145/1833272.1833274⟩ – reference download: 2012-02-01, BibWeb/PDF

O'Hara, Keith J. a. *Jennifer S. Kay*: Open source software and computer science education; in: J. Comput. Small Coll. 18 February (2003), pp. 1–7 ⟨URL: http://dl.acm.org/citation.cfm?id=771712.771716⟩, ISSN 1937–4771

Omsels, Hermann-Josef: Open Source und das deutsche Vertrags- und Urheberrecht; in: *Christian Schertz* a. *Herman-Josef Omsels, editors:* Festschrift für Paul W. Hertin zum 60. Geburtstag; 2000

Open Source Development Labs: OSDL announces patent common project; 2005 ⟨URL: http://www.osdl.org/newsroo,/press_releases/2005/2005_08_09_beaverton.html⟩

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

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

Open Source Initiative: Common Development and Distribution License (CDDL-1.0); 2004 [n.y. of the html page itself], FreeWeb/HTML ⟨URL: http://opensource.org/licenses/CDDL-1.0⟩ – reference download: 2013-04-19

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

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

Open Source Initiative: European Union Public License, version 1.1 (EUPL-1.1; 2007 [n.y. of the html page itself], FreeWeb/HTML ⟨URL: http://opensource.org/licenses/EUPL-1.1⟩ – reference download: 2013-03-04

Open Source Initiative: GNU Affero General Public License, Version 3 (AGPL-3.0); 2007 [n.y. of the html page itself], FreeWeb/HTML ⟨URL: http://opensource.org/licenses/AGPL-3.0⟩ – reference download: 2013-04-05

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

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

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

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

Open Source Initiative: The MIT License; 2012 [n.y.], FreeWeb/HTML ⟨URL: http://opensource.org/licenses/mit-license.php⟩ – reference download: 2012-08-24

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

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

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

Open Source Initiative: The [OSI] Licence Review Process; 2012 [n.y.], FreeWeb ⟨URL: http://www.opensource.org/approval⟩ – reference download: 2013-01-22

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

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

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

Open Source Initiative: Open Source Licenses by Category; 2013 [n.y.], FreeWeb ⟨URL: http://opensource.org/licenses/category⟩ – reference download: 2013-01-29

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

Open Source Initiative: The PostgreSQL Licence (PostgreSQL); 2013 [n.y.], FreeWeb/HTML ⟨URL: http://opensource.org/licenses/PostgreSQL⟩ – reference download: 2013-02-27

Oreg, Shaul a. *Oded Nov*: Exploring motivations for contributing to open source initiatives: The roles of contribution context and personal values; in: Computers in Human Behavior, 24 (2008), No. 5, pp. 2055–2073 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0747563207001537⟩ – reference download: 2012-02-01, BibWeb/PDF

O'Reilly, Tim: Lessons from Open-Source Software Development; in: Communications of the ACM, 42 (1999), No. 4, pp. 32–37 ⟨URL: http://doi.acm.org/10.1145/299157.299164⟩ – reference download: 2011-12-28, BibWeb/PDF

O'Reilly, Tim, editor: Open Source: kurz und gut; 1999

Orsila, Heikki et al.: Trust Issues in Open Source Software Development; in: Proceedings of the Warm Up Workshop for ACM/IEEE ICSE 2010; New York, NY, USA: ACM, 2009 (= WUP '09) ⟨URL: http://doi.acm.org/10.1145/1527033.1527037⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–60558–565–9, pp. 9–12

Osterloh, M., S. Rota, a. *M. von Wartburg*: Open Source - New Rules in Software Developement; Zürich, 2001, working paper ⟨URL: http://www.iou.unizh,ch/orga/downloads/OpenSourceAoM.pdf⟩

Osterloh, Margit a. *Sandra Rota*: - Just another case of collective invention? in: Research Policy, 36 (2007), No. 2, pp. 157–171 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0048733306001983⟩ – reference download: 2012-02-01, BibWeb/PDF

Osterloh, Margit, *Sandra Rota*, a. *Bernhard Kuster*: Open Source Software Production: Climbing on the Shoulders of Giants; 2002 ⟨URL: http://opensource.mit.edu/papers/osterlohrotakuster.pdf⟩

Osterloh, Margit, *Sandra Rota*, a. *Roger Lüthi*: 'Collective Invention' als neues Innovationsmodell; In *Drossou, Krempl, a. Poltmann*: Die wunderbare Wissensvermehrung, 2006, pp. 65–76, Print

O'Sullivan, Maureen: Eof[:] Free Software Licenses; in: Linux Journal, 122 June (2004), pp. Article No. 11 ⟨URL: http://dl.acm.org/citation.cfm?id=993247.993258⟩ – reference download: 2011-12-28, BibWeb/HTML

O'Mahony, Siobhán: Guarding the commons: how community managed software projects protect their work; in: RP, 32 (2003), No. 7, pp. 1179–1198 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0048733303000489⟩ – reference download: 2012-02-09, BibWeb/PDF

Patterson, Chip: Copyright Misuse and Modified Copyleft: New Solutions to the Challenges of Internet; in: Michigan Law Review, 98 (2000), No. 5, pp. 1351–1383, BibWeb/PDF

Pelizza, Annalisa: Openness as an Asset: A Classification System for Online Communities Based on Actor-Network Theory; in: Proceedings of the 6th International Symposium on Wikis and Open Collaboration; New York, NY, USA: ACM, 2010 (= WikiSym '10) ⟨URL: http://doi.acm.org/10.1145/1832772.1832784⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0056–8, pp. 8:1–8:10

Perens, Bruce: The Open Source Definition; In *DiBona, Ockman, a. Stone*: Open Sources, 1999, pp. 171–188

Perens, Bruce: Combining GPL and Proprietary Software; in: Datamation, 9 (2009), pp. wp. [3 pages] ⟨URL: http://www.datamation.com/osrc/article.php/3801396/Bruce-Perens-Combining-GPL-and-Proprietary-Software.htm⟩ – reference download: 2012-03-09, FreeWeb/HTML

Perr, Jon, *Melissa M. Appleyard*, a. *Patrick Sullivan*: Open for business: emerging business models in open source software; in: INTERNATIONAL JOURNAL OF TECHNOLOGY MANAGEMENT, 52 (2010), pp. 432–456

Peters, Stormy: Open Source Is Changing the Way Work Gets Done; conference contribution; In *Boldyreff et al.*: Open Source Ecosystems, 2009, p. 1, BibWeb/PDF

Petreley, Nicholas: /var/opinion: The GPLv2 vs. GPLv3 Debate; in: Linux Journal, 153 January (2007), p. 17 ⟨URL: http://dl.acm.org/citation.cfm?id=1194955.1194972⟩ – reference download: 2011-12-28, BibWeb/HTML

Phillips, Douglas E.: The Software License Unveiled. How Legislation by License Controls Software Access; Oxford, New York, Auckland [etc. ...]: Oxford University Press, 2009, ISBN 978–0–19–534187–4

Piller, Frank T.: User Innovation: der Kunde kann's besser; In *Drossou, Krempl, a. Poltmann*: Die wunderbare Wissensvermehrung, 2006, pp. 85–97, Print

Piller, Harald: Von Open Source zu Open Innovation; in: Harvard Business Manager, 25 (2003), No. 12, p. 114

Pisano, G.: Profiting from Innovation and the Intellectual Property Revolution; in: Research Policy, 35 (2006), No. 8, pp. 1122–1130

Plaß, Gunda: Open Contents im deutschen Urheberrecht; in: GRUR, (2002), pp. 670ff

Polanski, Arnold: Is the General Public Licence a Rational Choice? in: Journal of Industrial Economics, 55 (2007), pp. 691–714, BibWeb/PDF

Prechelt, Lutz: Some Non-Usage Data for a Distributed Editor: the Saros Outreach; in: Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering; New York, NY, USA: ACM, 2011 (= CHASE '11) ⟨URL: http://

doi.acm.org/10.1145/1984642.1984651⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 978–1–4503–0576–1, p. 48

Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development; (= FLOSS '09) Washington, DC, USA: IEEE Computer Society, 2009 ⟨URL: http://dl.acm.org/citation.cfm?id=1572192⟩ – reference download: 2012-01-25, BibWeb/PDF, ISBN 978–1–4244–3720–7

Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development; [General Chairs: Justin Erenkrantz and Hyrum K. Wright]; (= FLOSS '10) New York, NY, USA: ACM, 2010 ⟨URL: http://dl.acm.org/citation.cfm?id=1833272⟩ – reference download: 2012-01-25, BibWeb/PDF, ISBN 978–1–60558–978–7

Proceedings of the Fifth Workshop on Open Source Software Engineering; (= 5-WOSSE) New York, NY, USA: ACM, 2005 ⟨URL: http://doi.acm.org/10.1145/1082983.1083260⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–59593–127–9

Qureshi, Israr a. *Yulin Fang*: Socialization in Open Source Software Projects: A Growth Mixture Modeling Approach; in: Organizational Research Methods, 14 (2011), No. 1, pp. 208–238, BibWeb/PDF

Rafiq, Muhammad: LIS community's perceptions towards open source software adoption in libraries; in: International Information & Library Review (2009) 41, 137e145, 41 (2009), pp. 137–145, BIbWeb/PDF

Raja, Uzma a. *Evelyn Barry*: Investigating quality in large-scale Open Source Software; In Proceedings of the Fifth Workshop on Open Source Software Engineering, 2005, pp. 1–4 ⟨URL: http://doi.acm.org/10.1145/1082983.1083268⟩ – reference download: 2012-02-01, BibWeb/PDF

Raymond, Eric: A Brief History of Hackerdom, revised version; 2000 ⟨URL: http://www.catb.org/~esr/writings/hacker-history/hacker-history.html⟩

Raymond, Eric S.: How To Become A Hacker; ⟨URL: http://www.catb.org/esr/faqs/hacker-howto.html⟩

Raymond, Eric S.: The Cathedral and the Bazaar; in: First Monday, 3 March (1998), No. 3, p. o.A.

Raymond, Eric S.: Homesteading the Noosphere: An Introductory Contradiction; in: First Monday, 3 (1999), No. 10, p. o.A.

Raymond, Eric S.: The cathedral and the bazaar : musings on Linux and open source by an accidental revolutionary; Peking [...]: ???, 2001

Reed, Matthew W. et al.: Developing and Learning Web Services with Open Source Software: An Experience Report; in: JCSC, 22 April (2007), No. 4, pp. 93–100 ⟨URL: http://dl.acm.org/citation.cfm?id=1229637.1229654⟩ – reference download: 2011-12-29, BibWeb/PDF

Reese, Björn a. *Daniel Sternberg*: Working without Copyleft; 2001 ⟨URL: http://www.oreillynet.com/lpt/a/1403⟩

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

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

Reincke, Karsten, *Greg Sharpe*, a. *contributors*: Open Source License Compendium. How to Achieve Open Source License Compliance; 2015, FreeWeb/PDF ⟨URL: http://www.oslic.org/releases/oslic.pdf⟩ – reference download: 2015-01-20

Reitmayr, Gerhard a. *Dieter Schmalstieg*: An Open Software Architecture for Virtual Reality Interaction; in: Proceedings of the ACM Symposium on Virtual Reality Software and Technology; New York, NY, USA: ACM, 2001 (= VRST '01) ⟨URL: http://doi.acm.org/10.1145/505008.505018⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–58113–427–4, pp. 47–54

Renner, Thomas et al.: Open Source Software. Einsatzpotentiale und Wirtschaftlichkeit; eine Studien der Fraunhofer Gesellschaft; Stuttgart: Fraunhofer IRB Verlag, 2005, Print, ISBN 3–8167–7008–8

Reynolds, Carl J a. *Jeremy C Wyatt*: Open Source, Open Standards, and Health Care Information Systems; in: JMIR, 13 (2011), No. 1, p. wp ⟨URL: http://www.jmir.org/2011/1/e24/⟩, BibWeb/HTML

Rigby, Peter C., *Daniel M. German*, a. *Margaret-Anne Storey*: Open Source Software Peer Review Practices: A Case Study of the Apache Server; in: Proceedings of the 30th International Conference on Software Engineering; New York, NY, USA: ACM, 2008 (= ICSE '08) ⟨URL: http://doi.acm.org/10.1145/1368088.1368162⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 978–1–60558–079–1, pp. 541–550

Rigby, Peter C. a. *Margaret-Anne Storey*: Understanding Broadcast Based Peer Review on Open Source Software Projects; in: Proceedings of the 33rd International Conference on Software Engineering; New York, NY, USA: ACM, 2011 (= ICSE '11) ⟨URL: http://doi.acm.org/10.1145/1985793.1985867⟩ – reference download: 2011-12-29, BibWen/PDF, ISBN 978–1–4503–0445–0, pp. 541–550

Rivlin, Gary: Linus Torvalds - Leader of the Free World; in: Wired Magazin, (2003), No. 11, pp. 152ff

Robbins, Arnold: What's GNU? in: Linux Journal, 1 March (1994), p. 9:1 ⟨URL: http://dl.acm.org/citation.cfm?id=328204.328213⟩ – reference download: 2011-12-28, BibWeb/HTML

Robert W, Guomulkiewics: How Copyleft uses License Right to succeed in the OPen Source Software Revolution and the Implications fpr Article 2b; in: Houston Law Review, 36 (???), pp. 179ff

Roberts, Keith A.: Generic Methodology for Open Source Software Development; in: SIGSOFT Software Engineering Notes, 30 March (2005), No. 2, pp. 1–5 ⟨URL: http://doi.acm.org/10.1145/1050849.1050863⟩ – reference download: 2011-12-29, BibWeb/PDF

Rose, Marshall T.: The Open Book, A Practical Perspective on OSI; Englewood Cliffs NJ: Prentice Hall, 1990, Print, ISBN 0–13–643016–3

Rosen, Lawrence: Geek Law[:] A Question of Licenses; in: Linux Journal, 89 September (2001), p. 14 ⟨URL: http://dl.acm.org/citation.cfm?id=509824.509838⟩ – reference download: 2011-12-28, BibWeb/HTML

Rosen, Lawrence: Geek Law[:] Copyright Questions; in: Linux Journal, 88 August (2001), p. 13 ⟨URL: http://dl.acm.org/citation.cfm?id=509800.509813⟩ – reference download: 2011-12-28, BibWeb/HTML

Rosen, Lawrence: Geek Law[:] License FUD; in: Linux Journal, 92 December (2001), p. 14 ⟨URL: http://dl.acm.org/citation.cfm?id=512620.512634⟩ – reference download: 2011-12-29, BibWeb/HTML

Rosen, Lawrence: Geek Law[:] Naming Open-Source Software; in: Linux Journal, 90 October (2001), p. 11 ⟨URL: http://dl.acm.org/citation.cfm?id=509852.509863⟩ – reference download: 2011-12-28, BibWeb/HTML

Rosen, Lawrence: Geek Law[:] Allocation of the Risks; in: Linux Jo, 101 September (2002), p. 17 ⟨URL: http://dl.acm.org/citation.cfm?id=566949.566966⟩ – reference download: 2011-12-29, BibWeb/HTML

Rosen, Lawrence: Geek Law[:] Bad Law; in: Linux Journal, 98 June (2002), p. 13 ⟨URL:

http://dl.acm.org/citation.cfm?id=513489.513502⟩ – reference download: 2011-12-29, BibWeb/PDF

Rosen, Lawrence: Geek Law[:] Dealing with Patents in Softwar Licences; in: Linux Journal, 93 January (2002), p. 14 ⟨URL: http://dl.acm.org/citation.cfm?id=512788.512802⟩ – reference download: 2011-12-29, BibWeb/HTML

Rosen, Lawrence: Geek Law[:] Dealing With Patents in Software Licenses, Part II; in: Linux Journal, 94 February (2002), p. 15 ⟨URL: http://dl.acm.org/citation.cfm?id=513039.513054⟩ – reference download: 2011-12-28, BibWeb/HTML

Rosen, Lawrence: Geek Law[:] Fair Use; in: Linux Journal, 100 August (2002), p. 18 ⟨URL: http://dl.acm.org/citation.cfm?id=563953.563971⟩ – reference download: 2011-12-28, BibWeb/HTML

Rosen, Lawrence: Geek Law[:] License Defamation; in: Linux Journal, 99 July (2002), p. 15 ⟨URL: http://dl.acm.org/citation.cfm?id=513581.513596⟩ – reference download: 2011-12-29, BibWeb/HTML

Rosen, Lawrence: Geek Law[:] Unbiased License FUD; in: Linux Journal, 95 March (2002), p. 15 ⟨URL: http://dl.acm.org/citation.cfm?id=513085.513100⟩ – reference download: 2011-12-29, BibWeb/HTML

Rosen, Lawrence: Geek Law[:] Why the Public Domain Isn't a License; in: Linux Journal, 102 October (2002), p. 12 ⟨URL: http://dl.acm.org/citation.cfm?id=571785.571797⟩ – reference download: 2011-12-29, BibWeb/HTML

Rosen, Lawrence: IAAL[:] Derivative Works; in: Linux Journal, 105 January (2003), p. 13 ⟨URL: http://dl.acm.org/citation.cfm?id=603771.603784⟩ – reference download: 2011-12-28, BibWeb/HTML

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

Rosen, Lawrence: OSL 3.0: A Better License for Open Source Software; in: CRi, 6 (2007), pp. 166–171, Copy

Rosenberg, D. K.: Open source - The unauthorized white papers; Chicago, 2000

Rossi, Christina a. *Andrea Bonaccorsi*: Intrinsic motivations and preit-oriented firms supplying Open Source products and services; in: First Monday, 10 May (2005), No. 5, p. o.a.

Rossi, Cristina a. *Andrea Bonaccorsi*: Why profit-oriented companies enter the OS field?: Intrinsic vs. extrinsic incentives; In Proceedings of the Fifth Workshop on Open Source Software Engineering, 2005, pp. 12:1–12:5 ⟨URL: http://doi.acm.org/10.1145/1082983.1083269⟩ – reference download: 2011-12-29, BibWeb/PDF

Rossi, Naria Alessandra: Decoding the "Fre/Open Sour ce(F/OSS) Software Puzzle": a survey of theoretical and empirical contributions; 2004 ⟨URL: http://opensource.mit.edu/papers/rossi.pdf⟩

Ruffin, M. a. *C. Ebert*: Using Open Source Software in Product Development: A primer; in: IEEE SOFTWARE, 21 (2004), No. 1, pp. 82–86

Sabin, Mihaela: Free and Open Source Software Development of IT Systems; in: Proceedings of the 2011 Conference on Information Technology Education; New York, NY, USA: ACM, 2011 (= SIGITE '11) ⟨URL: http://doi.acm.org/10.1145/2047594.2047601⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–1017–8, pp. 27–32

Sakurai, Cledson Akio a. *Moacyr Martucci Junior*: An Open System Architecture for Operation Support System at Telecommunications Service Providers; in: Proceedings of the 1st International Symposium on Information and Communication Technologies; o.O.: Trinity College Dublin, 2003 (= (ISICT '03)) ⟨URL: http://dl.acm.org/citation.cfm?id=963600.963705⟩ – reference download: 2012-02-01, BibWeb/PDF, pp. 524–529

Samoladas, Ioannis et al.: Open Source Software Development Should Strive for Even Greater Code Maintainability; in: Communications of the ACM, 47 October (2004), No. 10, pp. 83–87

⟨URL: http://doi.acm.org/10.1145/1022594.1022598⟩ – reference download: 2011-12-29, BibWeb/PDF

Samuelson, Pamela: IBM's Pragmatic Embrace of Open Source; in: Communications of the ACM, 49 (2006), No. 10, pp. 21–25 ⟨URL: http://doi.acm.org/10.1145/1164394.1164412⟩ – reference download: 2011-12-28, BibWeb/PDF

Samuelson, Pamela: Legally Speaking[:] When is a "License" Really a Sale? in: Communications of the ACM, 52 March (2009), No. 3, pp. 27–29 ⟨URL: http://doi.acm.org/10.1145/1467247.1467258⟩ – reference download: 2011-12-29, BibWeb/PDF

Sandred, J.: Managing open source projects; New York, 2001

Santos Jr., Carlos Denner et al.: Intellectual Property Policy and Attractiveness: A Longitudinal Study of Free and Open Source Software Projects; in: Proceedings of the ACM 2011 conference on Computer supported cooperative work; New York, NY, USA: ACM, 2011 (= CSCW '11) ⟨URL: http://doi.acm.org/10.1145/1958824.1958950⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 978–1–4503–0556–3, pp. 705–708

Sauer, Robert M.: Why develop open-source software? The role of non-pecuniary benefits, monetary rewards, and open-source licence type; in: Oxford Review of Economic Policy, 23 (2007), No. 4, pp. 605–619, BibWeb/PDF

Sauerburger, Heinz, editor: Open Source Software; dpunkt.verlag, 2004

Savage, S. S.: Conquering Open Source Fears; in: Linux Executive Report, (2006) ⟨URL: http://www.ibm.com/linux/⟩

Scacchi, W.: Understanding the Requirements for Developing Open Source Software Systems; in: IEEE Procedings Software, 149 (2002), pp. 24–39

Scacchi, Walt: OpenEC/B: Electronic Commerce and Free/Open Source Software Development; In Proceedings of the Fifth Workshop on Open Source Software Engineering, 2005, pp. 8:1–8:5 ⟨URL: http://doi.acm.org/10.1145/1082983.1083270⟩ – reference download: 2011-12-29, BibWeb/PDF

Scacchi, Walt: Free/Open Source Software Development: Recent Research Results and Emerging Opportunities; in: The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering: companion papers; New York, NY, USA: ACM, 2007 (= ESEC-FSE companion '07) ⟨URL: http://doi.acm.org/10.1145/1295014.1295019⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–59593–812–1, pp. 459–468

Scacchi, Walt: The Future of Research in Free/Open Source Software Development; in: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research; New York, NY, USA: ACM, 2010 (= FoSER '10) ⟨URL: http://doi.acm.org/10.1145/1882362.1882427⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0427–6, pp. 315–320

Schiff, Aaron: The economics of open source software: A survey of the early literature; in: The Review of Network Economics, 1 March (2002), No. 1, pp. 66–74

Schiffner, Thomas: Open Source Software - Freie Software im deutschen Urheber- und Vertragsrecht; München, 2002

Schlesinger, David: Working with Open Source: A Practical Guide; in: interactions, 14 November/December (2007), pp. 35–37 ⟨URL: http://doi.acm.org/10.1145/1300655.1300678⟩ – reference download: 2011-12-29, BibWeb/PDF

Schmitz, L.: Linuxworld: Debatte um Pool für Open-Source-Patente; in: Computerwoche, (2005) ⟨URL: http://www.computerwoche.de/index.cfm?pageid=254\&artid=79815⟩

Schneider, Jochen: Handbuch des EDV-Rechts; 4th edition. Köln: Dr. Otto Schmidt, 2009

Schricker a. *Ulrich Loewenheim, editors*: Urheberrecht; Kommentar; 4th edition. München: Beck, 2010

Schryen, Guido: Is open source security a myth? in: Commununications of the ACM, 54 (2011), pp. 130–140 ⟨URL: http://doi.acm.org/10.1145/1941487.1941516⟩, BibWeb/PDF

Schryen, Guido a. *Rouven Kadura*: Open source vs. closed source software: towards measuring security; in: *Sung Y. Shin* a. *Sascha Ossowski, editors:* Proceedings of the 2009 ACM symposium on Applied Computing; New York, NY, USA: ACM, 2009 (= SAC '09) ⟨URL: http://doi.acm.org/10.1145/1529282.1529731⟩ – reference download: 2012-01-06, BibWeb/PDF, ISBN 978–1–60558–166–8, pp. 2016–2023

Schulz, Carsten: VSI-Gutachten zu Open-Source-Software. Die scharfe Klinge des Gesetzes? in: Linux-Magazin, (2003), pp. 68ff

Schulz, Carsten: Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte. Vertragsstrukturen in Open Source Modellen; Köln, 2005

Schäfer, Fabian: Der virale Effekt. Entwicklungsrisiken im Umfeld von Open Source Software; Karlsruhe: Universitätsverlag Karlsruhe, 2007, BibWeb/PDF, ISBN 978–3–86644–141–5

Searls, Doc: Linux for Suits: Linus Takes a Pass on the New GPL Draft; in: Linux Journal, 145 May (2006), p. 15 ⟨URL: http://dl.acm.org/citation.cfm?id=1134160.1134175⟩ – reference download: 2011-12-28, BibWeb/HTML

Searls, Doc: Eof[:] Why to Build on Foss in the First Place; in: Linux Journal, 165 January (2008), pp. Article No. 16 ⟨URL: http://dl.acm.org/citation.cfm?id=1344189.1344205⟩ – reference download: 2011-12-29, BibWeb/HTML

Searls, Doc: Eof[:] The Power of Definitions; in: Linux Journal, 177 January (2009), pp. Article No. 15 ⟨URL: http://dl.acm.org/citation.cfm?id=1502508.1502523⟩ – reference download: 2011-12-28, BibWeb/HTML

Sebald, Gerd: Offene Wissensökonomie. Analysen zur Wissenssoziologie der Free/Open Source Softwareentwicklung; Dissertation; Wiesbaden: VS Verlag für Sozialwissenschaften, 2008, Print a. BibWeb/PDF, ISBN 978–3–531–15705–4

Seel, Bernd a. *Miriam Kraft*: Einführung in das Prinzip Open Source; In *Asche et al.*: Open Source. Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen, 2008, pp. 9–19, Print

Seemayer, Walter a. *Jason Matusow*: Das Microsoft-Shared-Source-Programm aus der Business-Perspektive; In *Lutterbeck a. Bärwolff*: Open Source Jahrbuch, 2005, pp. 185–200 ⟨URL: http://...mitaufnehmen..⟩

Sen, Ravi, *Chandrasekar Subramaniam*, a. *Matthew Nelson*: Determinants of the Choice of Open Source Software License; in: J. Manage. Inf. Syst. 25 December (2008), pp. 207–240 ⟨URL: http://dl.acm.org/citation.cfm?id=1554453.1554460⟩, ISSN 0742–1222

Sen, Ravi, *Chandrasekar Subramaniam*, a. *Matthew L. Nelson*: Open source software licenses: Strong-copyleft, non-copyleft, or somewhere in between? in: Decision Support Systems, 52 (2011), No. 1, pp. 199–206 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0167923611001242⟩ – reference download: 2012-02-01, BibWeb/PDF

Sester, Peter: Open-Source-Software: Vertragsrecht, Haftungsrisiken und IPR-Fragen; in: CR [Computer und Recht], (2000), pp. 797ff

Sethanandha, Bhuricha Deen: Improving Open Source Software Patch Contribution Process: Methods and Tools; in: Proceedings of the 33rd International Conference on Software Engineering; New York, NY, USA: ACM, 2011 (= ICSE '11) ⟨URL: http://doi.acm.org/10.1145/1985793.1986018⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 978–1–4503–0445–0, pp. 1134–1135

Shibuya, Bianca a. *Tetsuo Tamai*: Understanding the Process of Participating in Open Source Communities; In Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009, pp. 1–6 ⟨URL: http://dx.doi.org/10.1109/FLOSS.2009.5071352⟩ – reference download: 2012-02-01, BibWeb/PDF

Siepmann, Jürgen: Lizenz- und haftungsrechtliche Fragen bei der kommerziellen Nutzung Freier Software; in: JurPC Web-Dok, (1999), p. 163

Siepmann, Jürgen: Freie Software - Rechtfreier Raum? Rechtssicherheit im Umgang mit Open Source Software; München, 2000

Singh, Param Vir: The Small-World Effect: The Influence of Macro-Level Properties of Developer Collaboration Networks on Open-Source Project Cuccess; in: Transactions on Software Engineering Methodology, 20 (2010), No. 2, pp. 6:1–6:27 ⟨URL: http://doi.acm.org/10.1145/1824760.1824763⟩ – reference download: 2011-12-29, BibWeb/PDF

Siponen, Mikko: A Justification for Software Rights; in: SIGCAS, 36 September (2006), No. 3, pp. 11–20 ⟨URL: http://dl.acm.org/citation.cfm?id=1195716.1195718⟩ – reference download: 2011-12-29, BibWeb/PDF

Sirkkala, Petri, *Timo Aaltonen*, a. *Imed Hammouda*: Opening Industrial Software: Planting an Onion; conference contribution; In *Boldyreff et al.*: Open Source Ecosystems, 2009, pp. 57–69, BibWeb/PDF

Smith, Bradford L.: The Future of Software: Enabling the Marketplace to Decide; In *Hahn*: Government Policy toward Open Source Software, 2002

Sojer, Manuel: Reusing Open Source Code. Value Creation and Value Appropriation. Perspectives on Knowledge Reuse; wuth a Foreword by Univ.-Prof. Dr. Joachim Henkel; Wiesbaden: Gabler, 2011 (= Gabler Research[:] Innovation und Entrepreneurship), BibWEB/PDF, ISBN 978–8349–2668–5

Sojer, Manuel a. *Joachim Henkel*: License Risks from Ad Hoc Reuse of Code from the Internet; in: Communications of the ACM, 54 December (2011), No. 12, pp. 74–81 ⟨URL: http://doi.acm.org/10.1145/2043174.2043193⟩ – reference download: 2011-12-28, BibWeb/PDF

Soto, Martin a. *Marcus Ciolkowski*: The QualOSS Open Source Assessment Model Measuring the Performance of Open Source Communities; in: Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement; Washington, DC, USA: IEEE Computer Society, 2009 (= ESEM '09) ⟨URL: http://dx.doi.org/10.1109/ESEM.2009.5314237⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 978–1–4244–4842–5, pp. 498–501

Sowe, Sulayman K., *Ioannis Stamelos*, a. *Lefteris Angelis*: Understanding knowledge sharing activities in free/open source software projects: An empirical study; in: Journal of Systems and Software, 81 (2008), No. 3, pp. 431–446 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0164121207000842⟩ – reference download: 2012-02-03, BibWeb/PDF

Spielkamp, Mathias: Creative Commons[:] Andere Zeiten, andere Lizenzen; In *Djordjevic et al.*: Urheberrecht im Alltag, 2008, pp. 219–221, Print

Spielkamp, Mathias: Lessigletters-Remix[:] Die Creative-Commons-Initiative; In *Djordjevic et al.*: Urheberrecht im Alltag, 2008, pp. 223–230, Print

Spindler, Gerald: Rechtsfragen der Open Source Software. Gutachten im Auftrags des VSI; München, 2003

Spindler, Gerald: Stellungnahme [zum Gutachten der VSI]; in: Linux-Magazin, (2003), No. 9, p. 70

Spindler, Gerald: Ausgewählte urheberrechtliche Problem von Open Source Software unter der GPL; in: *Alfread Büllesbacg* a. *Thomas Dreier, editors:* Wem gehört die Information im 21. Jahrhundert; 2004

Spindler, Gerald: Open Source Software auf dem gerichtlichen Prüfstand - Dingliche Qualifikation und Inhaltskontrolle; in: Kommunikation und Recht, (2004), pp. 528–524

Spindler, Gerald; Spindler, Gerald, editor: Rechtsfragen bei Open Source Software; Köln: Verlag Dr. Otto Schmidt KG, 2004, Print, ISBN 3–504–56080–0

Spindler, Gerald a. *Andreas Wiebe*: Open Source-Vertrieb - Rechteeinräumung und Nutzungsberechtigung; in: Computerrecht, (2003), pp. 873–879

Splittgerber, Andrea; Schröder, Georg F., editor: Lizenzen und Open Source rechtlicht einwandfrei nutzen. Eine klare Darstellung der Lizenzierung, Nutzungsrechtseinräumung und deren Auswirkung auf Vertragsgestaltung; Kissing: Weka Media, 2005, Print, ISBN 3–8245–1286–3

St. Laurent, Andrew N.: Understanding open source and free software licensing: guide to navigating licensing issues in existing & new software; Beijing and Köln: O'Reilly, 2004

Staff, CACM: True seeds of open source software; in: Commun. ACM, 52 January (2009), pp. 6–6 ⟨URL: http://doi.acm.org/10.1145/1435417.1435420⟩, ISSN 0001–0782

Stahl, Matthew T.: Open-source software: not quite endsville; in: Drug Discovery Today, 10 (2005), No. 3, pp. 219–222 ⟨URL: http://www.sciencedirect.com/science/article/pii/S1359644604033641⟩ – reference download: 2012-02-09, BibWeb/PDF

Stallman, Richard: Viewpoint: Why We Must Fight UCITA; in: Communications of the ACM, 43 June (2000), No. 6, pp. 27–28 ⟨URL: http://doi.acm.org/10.1145/336460.336470⟩ – reference download: 2011-12-29, BibWeb/PDF

Stallman, Richard: Can Freedom Withstand E-Books? in: Communications of the ACM, 44 March (2001), No. 3, p. 111 ⟨URL: http://doi.acm.org/10.1145/365181.365227⟩ – reference download: 2011-12-29, BibWeb/PDF

Stallman, Richard: Viewpoint: Why "Open Source" Misses the Point of Free Software; in: Commununications 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

Stallman, Richard M.: The Danger of Software Patents; 2001, FreeWeb/HTML ⟨URL: http://www.gnu.org/philosophy/stallman-mec-india.html⟩ – reference download: 2013-02-18

Stallman, Richard M.: Can You Trust Your Computer? [originally written in 2002]; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, pp. 115–117, Print

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, Richard M.: Free Software Definition; originally written in 1996; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, pp. 41–43, Print

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

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

Stallman, Richard M.: Free Software Needs Free Documentation; originally written in 2000; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, pp. 67–68, Print

Stallman, Richard M.: The GNU Manifesto; originally written in 1984; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, pp. 31–39, Print

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

Stallman, Richard M.: The Right to Read; originally written in 1997; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, pp. 73–77, Print

Stallman, Richard M.: Selling Free Software; originally written in 1996; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, pp. 63–65, Print

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

Stallman, Richard M.: What's in a Name? originally written in 2000; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, pp. 51–53, Print

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

Stallman, Richard M.: Why Software Should Not Have Owners; originally written in 1994; In *Stallman*: Free Software, Free Society: Selected Essays, 2002, pp. 45–49, Print

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.

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

Steinbring, Marc a. *Thorsten Hampel*: Connecting Babbling Bazaars - Der Open-Source-Gedanke im Wandel zum offenen Service; In *Asche et al.*: Open Source. Kommerzialisierungsmöglichkeiten und Chancen für die Zusammenarbeit von Hochschulen und Unternehmen, 2008, pp. 73–97, Print

Stewart, Katherine J., *Anthony P. Ammeter*, a. *Likoebe M. Maruping*: Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects; in: Informations Systems Research, 17 (2006), No. 2, pp. 126–144, BibWeb/PDF

Stewart, Katherine J., *David P. Darcy*, a. *Sherae L. Daniel*: Observations on Patterns of Development in Open Source Software Projects; In Proceedings of the Fifth Workshop on Open Source Software Engineering, 2005, pp. 1–5 ⟨URL: http://doi.acm.org/10.1145/1082983.1083272⟩ – reference download: 2011-12-29, BibWeb/PDF

Stol, Klaas-Jan a. *Muhammad Ali Babar*: Challenges in Using Open Source Software in Product Development: A Review of the Literature; [General Chairs: Justin Erenkrantz and Hyrum K. Wright]; In Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 2010, pp. 17–22 ⟨URL: http://doi.acm.org/10.1145/1833272.1833276⟩ – reference download: 2011-12-29, BibWeb/PDF

Stol, Klaas-Jan et al.: The Use of Empirical Methods in Open Source Software Research: Facts, Trends and Future Directions; In Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009, pp. 19–24 ⟨URL: http://dx.doi.org/10.1109/FLOSS.2009.5071355⟩ – reference download: 2012-02-01, BibWeb/PDF

Subramaniam, Chandrasekar, *Ravi Sen*, a. *Matthew L. Nelson*: Determinants of open source software project success: A longitudinal study; in: Decision Support System, 46 (2009), pp. 576–585, BibWeb/PDF

Subramanyam, Ramanath a. *Mu Xia*: Free/Libre Open Source Software development in developing and developed countries: A conceptual framework with an exploratory study; in: Decision Support Systems, 46 (2008), No. 1, pp. 173–186 ⟨URL: http://www.sciencedirect.com/science/article/pii/S016792360800119X⟩ – reference download: 2012-02-03, BibWeb/PDF

Suchomski, Bernd: Proprietäres Patentrecht beim Einsatz von Open Source Software. Eine rechtliche Analyse aus unternehmerischer Sicht; Bonn: Tgramedia, 2011 (= Medien Internet und Recht, [Vol./No.] 3), Print, ISBN 978–3–941192–03–4

Sujecki, Bartosz: Open Source Software im deutschen Vetrags- und Urheberrecht; in: Medien und Recht, (2005), pp. 40–48

Sweet, David: Andamooka: Open Support for Open Content; in: Linux Journal, 82 February (2001), pp. Article No. 13 ⟨URL: http://dl.acm.org/citation.cfm?id=364716.364729⟩ – reference download: 2011-12-28, BibWeb/HTML

Syme, Serena a. *L. Jean Camp*: The Governance of Code: Open Land vs. UCITA Land; in: SIGCAS, 32 (2002), No. 3, p. 2 ⟨URL: http://doi.acm.org/10.1145/644618.644623⟩ – reference download: 2011-12-29, BibWeb/HTML

Taubert, Niels C.: Produktive Anarchie? Netzwerke freier Softwareentwicklung; Bielefeld: transcript, 2006 (= Science Studies), Print, ISBN 3–89942–418–2

Terry, Michael, *Matthew Kay*, a. *Ben Lafreniere*: Perceptions and Practices of Usability in the Free/Open Source Software (FoSS) Community; in: Proceedings of the 28th International Conference on Human Factors in Computing Systems; New York, NY, USA: ACM, 2010 (= CHI '10) ⟨URL: http://doi.acm.org/10.1145/1753326.1753476⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–60558–929–9, pp. 999–1008

Terry, Michael et al.: ingimp: Introducing Instrumentation to an End-User Open Source Application; in: Proceedings of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems; New York, NY, USA: ACM, 2008 (= CHI '08) ⟨URL: http://doi.acm.org/10.1145/1357054.1357152⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 978–1–60558–011–1, pp. 607–616

Teupen, Christian: 'Copyleft' im deutschen Urheberrecht; Implikationen von Open Source Software im Urhebergesetz; Berlin: Duncker & Humblot, 2007 (= Schriften zum Bürgerlichen Recht, [Vol./No.] 367), Print, ISBN 978–3–428–12325–4

The Linux Foundation: SPDX License List; 2013, FreeWeb/HTML ⟨URL: http://spdx.org/licenses/⟩ – reference download: 2014-03-14

Themelidis, Markos: Open Source : die Freiheitsvision der Hacker; Frankfurt a.M.: ???, 2004

Theunissen, W. H. Morkel, *Andrew Boake*, a. *Derrick G. Kourie*: In Search of the Sweet Spot: Agile Open Collaborative Corporate Software Development; in: Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries; Republic of South Africa: South African Institute for Computer Scientists and Information Technologists, 2005 (= SAICSIT '05) ⟨URL: http://dl.acm.org/citation.cfm?id=1145675.1145705⟩ – reference download: 2011-12-28, BibWeb/PDF, ISBN 1–59593–258–5, pp. 268–277

Thorvalds, Linus: Just for fun : wie ein Freak die Computerwelt revolutionierte; die Biographie des Linux-Erfinders; München: ???, 2004

Torkar, Richard, *Pau Minoves*, a. *Janina Garrigós*: Adopting Free/Libre/Open Source Software. Practices, Techniques and Methods for Industrial Use; in: JAIS, 12 (2011), No. 1, pp. 88–122, BibWeb/PDF

Torvald, Linus: Just for Fun: wie ... [Biographie]; ???, 2004

Tsai, John: For Better or Worse: Introducing the GNU General Public License Version 3; in: Berkeley Technology Law Review, 23 (2008), pp. 547–581, Copy

Tsunoda, Masateru et al.: Analyzing OSS Developers' Working Time Using Mailing Lists Archives; in: Proceedings of the 2006 International Workshop on Mining Software Repositories; New York, NY, USA: ACM, 2006 (= MSR '06) ⟨URL: http://doi.acm.org/10.1145/1137983.1138031⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 1–59593–397–2, pp. 181–182

Turner, David: Anatomy of GPL Violations; in: Free Software Foundation Bulletin, (2002), No. 1, pp. 2–3

Turner, David: The LGPL and Java; 2004, FreeWeb/HTML ⟨URL: http://www.gnu.org/licenses/lgpl-java.en.html⟩ – reference download: 2015-02-09

Tuunanen, Timo, *Jussi Koskinen*, a. *Tommi Kärkkäinen*: Automated software license analysis; in: Automated Software Engineering, 16 (2009), pp. 455–490, BibWeb/PDF

Twidale, Michael: Silver Bullet or Fool's Gold: Supporting Usability in Open Source Software Development; in: Proceedings of the 27th International Conference on Software Engineering;

New York, NY, USA: ACM, 2005 (= ICSE '05) ⟨URL: http://doi.acm.org/10.1145/1062455.1062468⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 1–58113–963–2, p. 35

Uhr, Wolfrgang, *Werner Esswein*, a. *Eric Schoop, editors*: Wirtschaftsinformatik 2003 / Band II. Medien - Märkte - Mobilität; Heidelberg: Physica-Verlag, 2003

Välimäki, Mikko: Copyleft Licensing and EC Competition Law; in: E.C.L.R, 27 (2006), No. 3, pp. 130–136, Copy

Välimäki, Mikko, *Ville Oksanen*, a. *Juha Laine*: An Empirical Look at the Problems of Open Source Adoption in Finnish Municipalities; in: Proceedings of the 7th International Conference on Electronic Commerce; New York, NY, USA: ACM, 2005 (= ICEC '05) ⟨URL: http://doi.acm.org/10.1145/1089551.1089643⟩ – reference download: 2011-12-29, ISBN 1–59593–112–0, pp. 514–520

Valkov, Svilen: Innovative Concept of Open Source Enterprise Resource Planning (ERP) System; in: Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing; New York, NY, USA: ACM, 2008 (= CompSysTech '08) ⟨URL: http://doi.acm.org/10.1145/1500879.1500893⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–954–9641–52–3, pp. 11.6.1–11.6.7

Vamplew, Peter a. *Julian Dermoudy*: An Anti-Plagiarism Editor for Software Development Courses; in: *Alison Young* a. *Denis Tolhurst, editors:* Australasian Computing Education Conference; Necastle (Australia), 2005 (= Australia. Conferences in Researhc an Practice in Information Technology, [Vol./No.] 42), BibWeb/PDF, pp. 83–90

Van den Brande, Ywain, *Shane Coughlan*, a. *Till Jaeger, editors*: The International Free and Open Source Software Law Book; Munich (Germany): Open Source Press, 2011, Print, ISBN 978–3–941841–49–9

van Wendel de Joode, R., *J. A. de Bruijn*, a. *M. J. G. van Eeten*: Protecting the Virtual Commons. Self-Organizing Open Source and Free Software Communities and Innovative Intellectual Property Regimes; The Hague: T.M.C. Asser Press, 2003 (= Information Technology & Law Series, [Vol./No.] 3), Print, ISBN 90–6704–159–9

Ven, Kris a. *Jan Verelst*: The Importance of External Support in the Adoption of Open Source Server Software; conference contribution; In *Boldyreff et al.*: Open Source Ecosystems, 2009, pp. 116–128, BibWeb/PDF

Vetter, Greg R.: 'Infectious' Open Source Software: Spreading Incentives or Promoting Resistance? in: Rutgers Law Journal, 36:53 (2005), pp. 53–162

Viesel, Edward: Freiheit statt Freibier. Geschichte und Praxis der freien digitalen Welt - mit einer Einführung in Linux; Münster: Unrast-Verlag, 2006, Print, ISBN 3–897771–450–7

von Hippel, Eric a. *Georg von Krogh*: Open source software and the private-collective innovation model: Issues for organization science; in: Organization Science, 14 (2002), No. 2, pp. 209–223

von Hippel, Eric a. *Georg von Krogh*: The Promise of Research on Open Source Software; in: Management Science, 52 (2006), No. 7, pp. 975–983

von Krogh, G. a. *E. von Hippel*: The Promise of Research on Open Source Software; in: Management Science, 2006 (52), pp. 975–983

von Krogh, Georg a. *Sebastian Spaeth*: The open source software phenomenon: Characteristics that promote research; in: Journal of Strategic Information Systems, 16 (2007), p. 236–253, BibWeb/PDF

Välimäki, Mikko: The Rise of Open Source Licensing; A Challenge to the Use of Intellectual Property in the Software Industry;, PhD thesis ⟨URL: http://pub.turre.com⟩

Välimäki, Mikko: Dual Licensing in Open Source Software Industry; 2003 ⟨URL: http://opensource.mit.edu/papers/valimaki.pdf⟩

Wang, Yi, *Defeng Guo*, a. *Huihui Shi*: Measuring the Evolution of Open Source Software Systems with their Communities; in: SIGSOFT Software Engineering Notes, 32 November

(2007), No. 6, pp. 1–7 ⟨URL: http://doi.acm.org/10.1145/1317471.1317479⟩ – reference download: 2011-12-29, BibWeb/PDF

Watson, Richard T. et al.: The Business of Open Source; in: Communications of the ACM, 51 (2008), No. 4, pp. 41–46 ⟨URL: http://doi.acm.org/10.1145/1330311.1330321⟩

Weber, S.: The Success of Open Source; Cambridge MA: Harvard University Press, 2004

Weiss, Aaron: The Politics of Free (Software); in: netWorker, 5 September (2001), pp. 26–31 ⟨URL: http://doi.acm.org/10.1145/383719.383727⟩

Weiss, Michael: Economics of Collectives; in: Proceedings of the 15th International Software Product Line Conference; Volume 2, New York, NY, USA: ACM, 2011 ⟨URL: http://doi.acm.org/10.1145/2019136.2019181⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0789–5, pp. 39:1–39:8

West, Joel: How open is open enough? Melding proprietary and open source platform strategies; in: Research Policy, 32 (2003), pp. 1259–1285

Wheeler, David A.: Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers! 2002 ⟨URL: http://www.dwheeler.com/oss_fs_why.html⟩

Wichmann, Thorsten: Linux- und Open-Source-Strategien; Berlin, Heidelberg and New York: Springer, 2005, BibWeb/PDF, ISBN 3–540–22810–1

Widmer, Mike J.: Open Source Software - Urheberrechtliche Aspekte freier Software; Dissertation; Bern: Stämpfli Verlag, 2003, Print

Wiebe, A.: Softwarepatente und Open Source; in: CR [Computer und Recht], 20 (2004), No. 12, pp. 881–888

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

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

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

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

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

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

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

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

Witzel, Michaela: AGB-Recht und Open Source Lizenzmodelle; in: ITRB (IT-Rechtsberater), (2003), pp. 175ff

Wolf, M., K. Miller, a. *F. Grodzinsky*: On the meaning of free software; in: Ethics and Information Technology, 11 (2009), pp. 279–286 ⟨URL: http://dx.doi.org/10.1007/s10676-009-9207-9⟩ – reference download: 2012-02-03, BibWeb/PDF

Wolf, Marty J. et al.: Open Source Software: Intellectual Challenges to the Status Quo; in: Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education; New York, NY, USA: ACM, 2002 (= SIGCSE '02) ⟨URL: http://doi.acm.org/10.1145/563340.563464⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 1–58113–473–8, pp. 317–318

Wolf, Marty J., *Keith W. Miller*, a. *Frances S. Grodzinsky*: Free, Source-Code-Available, or Proprietary: An Ethically Charged, Context-Sensitive Choice; in: SIGCAS, 39 June (2009), No. 1, pp. 15–26 ⟨URL: http://doi.acm.org/10.1145/1565795.1565797⟩

Wolfe, Alexander: Toolkit: GNU Tools: Still Relevant? in: Queue, 1 December / January (2003/2004), pp. 14–17 ⟨URL: http://doi.acm.org/10.1145/966789.966795⟩ – reference download: 2011-12-29, BibWeb/PDF

Wuermeling, Ulrich a. *Thies Deike*: Open Source Software: Eine juristische Risikoanalyse; in: CR [Computer und Recht], (2003), pp. 87ff

Wynants, Marleen a. *Jan Cornelius, editors*: How Open is the Future? Economic, Social & Cultural Scenarios inspired by Free & Open-Source Software; Brüssel: VUB Brussels University Press, 2005

Xing, Guangming: Teaching Software Engineering Using Open Source Software; in: Proceedings of the 48th Annual Southeast Regional Conference; New York, NY, USA: ACM, 2010 (= ACM SE '10) ⟨URL: http://doi.acm.org/10.1145/1900008.1900085⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0064–3, pp. 57:1–57:3

Xu, Bo, *Donald R. Jones*, a. *Bingjia Shao*: Volunteers' involvement in online community based software development; in: Information & Management, 46 (2009), pp. 151–158, BibWeb/PDF

Yamakami, Toshihiko: Foundation-based Mobile Platform Software Engineering: Implications to Convergence to Open Source Software; in: Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human; New York, NY, USA: ACM, 2009 (= ICIS '09) ⟨URL: http://doi.acm.org/10.1145/1655925.1655962⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–60558–710–3, pp. 206–211

Yamakami, Toshihiko: OSS as a digital ecosystem: A Reference Model for Digital Ecosystem of OSS; in: Proceedings of the International Conference on Management of Emergent Digital EcoSystems; New York, NY, USA: ACM, 2010 (= MEDES '10) ⟨URL: http://doi.acm.org/10.1145/1936254.1936291⟩ – reference download: 2011-12-29, BibWeb/PDF, ISBN 978–1–4503–0047–6, pp. 207–208

Yatani, Koji et al.: Understanding How and Why Open Source Contributors Use Diagrams in the Development of Ubuntu; in: Proceedings of the 27th International Conference on Human Factors in Computing Systems; New York, NY, USA: ACM, 2009 (= CHI '09) ⟨URL: http://doi.acm.org/10.1145/1518701.1518853⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 978–1–60558–246–7, pp. 995–1004

Ye, Yunwen a. *Kouichi Kishida*: Toward An Understanding of the Motivation of Open Source Software Developers; in: Proceedings of the 25th International Conference on Software Engineering; Washington, DC, USA: IEEE Computer Society, 2003 (= ICSE '03) ⟨URL: http://dl.acm.org/citation.cfm?id=776816.776867⟩, ISBN 0–7695–1877–X, pp. 419–429

Yildirim, Nihan a. *Hacer Ansal*: Foresighting FLOSS (free/libre/open source software) from a developing country perspective: The case of Turkey; in: Technovation, 31 (2011), No. 12, pp. 666 – 678 ⟨URL: http://www.sciencedirect.com/science/article/pii/S0166497211001052⟩, ISSN 0166–4972

Yue, Kwok-Bun et al.: The Use of Free and Open Source Software in Real-World Capstone Projects; in: JCSC, 26 April (2011), No. 4, pp. 85–92 ⟨URL: http://dl.acm.org/citation.cfm?id=1953573.1953587⟩ – reference download: 2011-12-29, BibWeb/PDF

Yue, Kwok-Bun et al.: Open Courseware and Computer Science Education; in: JCSC, 20 October (2004), No. 1, pp. 178–186 ⟨URL: http://dl.acm.org/citation.cfm?id=1040231.1040255⟩

Zacchiroli, Stefano: Debian: 18 years of free software, do-ocracy, and democracy; in: Proceedings of the 2011 Workshop on Open Source and Design of Communication; New York, NY, USA:

ACM, 2011 (= OSDOC '11) ⟨URL: http://doi.acm.org/10.1145/2016716.2016740⟩, ISBN 978–1–4503–0873–1, pp. 87–87

Zhang, Wen, *Ye Yang*, a. *Qing Wang*: Network Analysis of OSS Evolution: An Empirical Study on ArgoUML Project; in: Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution; New York, NY, USA: ACM, 2011 (= IWPSE-EVOL '11) ⟨URL: http://doi.acm.org/10.1145/2024445.2024459⟩ – reference download: 2012-02-01, BibWeb/PDF, ISBN 978–1–4503–0848–9, pp. 71–80

Zhou, Ying a. *Joseph Davis*: Open source software reliability model: an empirical approach; in: Proceedings of the fifth workshop on Open source software engineering; New York, NY, USA: ACM, 2005 (= 5-WOSSE) ⟨URL: http://doi.acm.org/10.1145/1082983.1083273⟩, ISBN 1–59593–127–9, pp. 1–6

Zittrain, Jonathan: Normative Principles for Evaluating Free and Proprietary Software; in: University of Chicago Law Review, 71 (2004), No. 1, pp. 265–287, BibWeb/PDF

Zucker, William A.: Intellectual Property and Open Source: Copyright, Copyleft, and Other Issues for the User Community; in: Cutter IT Journal, 16 (2003), No. 5, pp. 27–34, Copy