FOSS Licensing

Part 1 – Basics & how to mark your own code

Matija Šuklje

September 2022

University of Maribor, Open Science Summer School

plan

plan

what we will look into today

Agenda

Part 1: basics & marking your own code

1. “IP” rights basics
2. typical use cases
3. common FOSS licenses in detail
4. how to mark your own code
5. REUSE.software demo

(15’ break)

Part 2: inbound licensing

“IP” rights

“IP” rights

commonalities

What is common to all “Intellectual Proprety” rights

exclusive rights to its holder[1]

others need a license (statutory, implict or express)

[1] Author (in ©) or inventor (in Pat.) may not be the ones holding the rights, if their rights were assigned.

“IP” rights

the big 3

Main “Intellectual Property” rights

copyright

automatic

expression (not ”mere idea”)

life + 70 years

since “fixation”

global

original

Main “Intellectual Property” rights

copyright	**patent**
automatic	registered
expression (not ”mere idea”)	invention (not abstracts)
life + 70 years	typically 20 years
since “fixation”	since filing of registration
global	per country
original	novel, inventive step, industrial applicability

Main “Intellectual Property” rights

copyright	**patent**	**trade marks**
automatic	registered	® registered, ™ unregistered
expression (not ”mere idea”)	invention (not abstracts)	brand/origin
life + 70 years	typically 20 years	10 years, renewable
since “fixation”	since filing of registration	since registration/use
global	per country	per country & class
original	novel, inventive step, industrial applicability	distinctive, non-generic
		need to enforce

“IP” rights

copyright

Copyright: How to obtain the rights

a. you = author
b. CAA/CLA
c. **(inbound) license**
d. (limited) use via ZASP

Copyright – In & Out

Inbound license

rights (in the code) you obtain from your upstream

Copyright – In & Out

Inbound license

rights (in the code) you obtain from your upstream

Outbound license

rights (in the code) you give or forward to your downstream

typical use cases

typical use cases

quick overview

Use cases: Quick overview

traditional/desktop distribution

- desktop software
- mobile apps
- embedded
- (OTA updates)
- client-side JS even in SaaS(!)

SaaS

- SaaS w/ user interface (i.e. front-end)
- SaaS w/o user interface (i.e. back-end)

FOSS

FOSS

basics

FOSS basics

- **use** the code
- **study** what the code does
- **share** the original code
- **improve** the code and share your modified version

“Free Software” = “Open Source Software” = “Libre Software” = “FOSS”

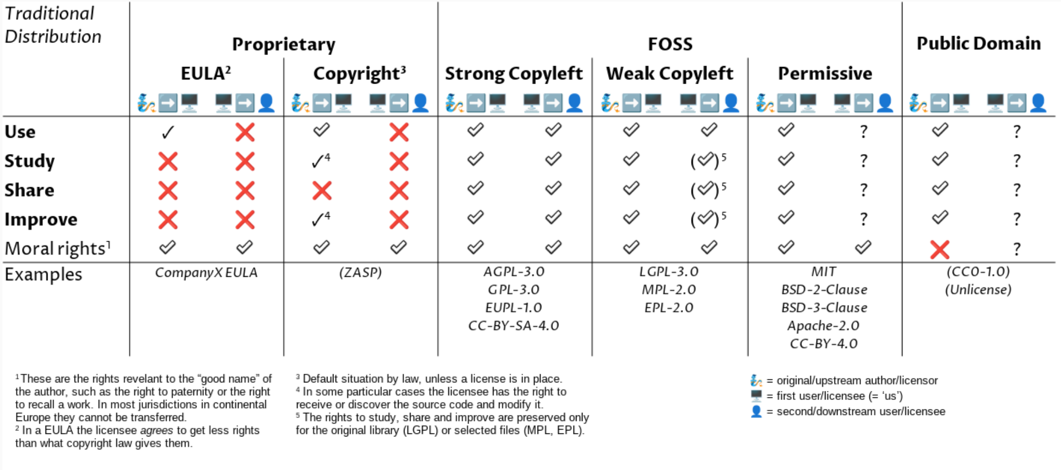
FOSS[2] != freeware

[2] “Free as in freedom, not free as in beer.”

FOSS

FOSS license overview

FOSS licenses – the traditional/desktop approach



Traditional Distribution	**Proprietary**				**FOSS**						**Public Domain**	
	EULA[2]		**Copyright**[3]		**Strong Copyleft**		**Weak Copyleft**		**Permissive**			
	🧞➡️🖥️	🖥️➡️👤	🧞➡️🖥️	🖥️➡️👤	🧞➡️🖥️	🖥️➡️👤	🧞➡️🖥️	🖥️➡️👤	🧞➡️🖥️	🖥️➡️👤	🧞➡️🖥️	🖥️➡️👤
Use	✓	❌	✔	❌	✔	✔	✔	✔	✔	?	✔	?
Study	❌	❌	✓[4]	❌	✔	✔	✔	(✔)[5]	✔	?	✔	?
Share	❌	❌	❌	❌	✔	✔	✔	(✔)[5]	✔	?	✔	?
Improve	❌	❌	✓[4]	❌	✔	✔	✔	(✔)[5]	✔	?	✔	?
Moral rights[1]	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	❌	?
Examples	*CompanyX EULA*		*(ZASP)*		*AGPL-3.0* *GPL-3.0* *EUPL-1.0* *CC-BY-SA-4.0*		*LGPL-3.0* *MPL-2.0* *EPL-2.0*		*MIT* *BSD-2-Clause* *BSD-3-Clause* *Apache-2.0* *CC-BY-4.0*		*(CC0-1.0)* *(Unlicense)*	

[1] These are the rights revelant to the "good name" of the author, such as the right to paternity or the right to recall a work. In most jurisdictions in continental Europe they cannot be transferred.

[2] In a EULA the licensee *agrees* to get less rights than what copyright law gives them.

[3] Default situation by law, unless a license is in place.

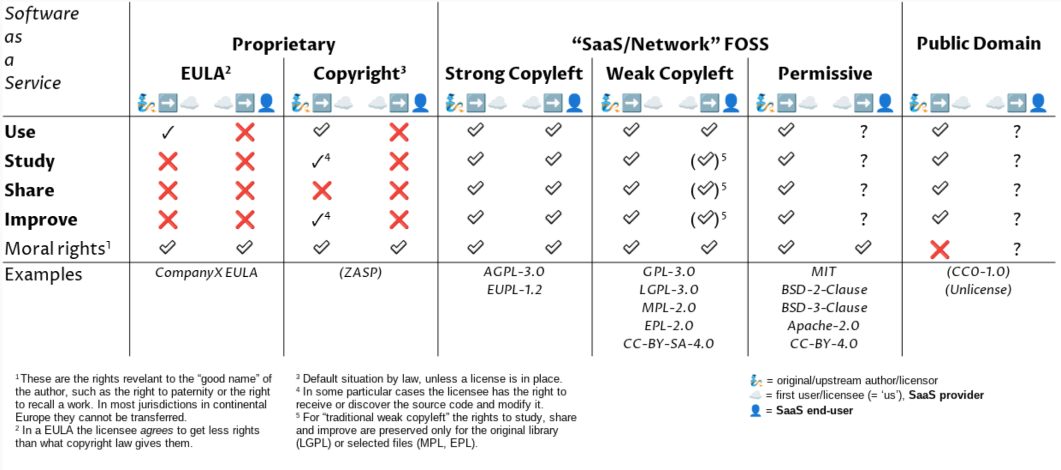
[4] In some particular cases the licensee has the right to receive or discover the source code and modify it.

[5] The rights to study, share and improve are preserved only for the original library (LGPL) or selected files (MPL, EPL).

🧞	= original/upstream author/licensor
🖥️	= first user/licensee (= 'us')
👤	= second/downstream user/licensee

Figure 1: FOSS licenses in traditional distribution model

FOSS licenses – the SaaS approach



Software as a Service	**Proprietary**				**“SaaS/Network” FOSS**						**Public Domain**	
	EULA[2]		**Copyright[3]**		**Strong Copyleft**		**Weak Copyleft**		**Permissive**			
	🧞➡☁	☁➡👤	🧞➡☁	☁➡👤	🧞➡☁	☁➡👤	🧞➡☁	☁➡👤	🧞➡☁	☁➡👤	🧞➡☁	☁➡👤
Use	✓	❌	✔	❌	✔	✔	✔	✔	✔	?	✔	?
Study	❌	❌	✓[4]	❌	✔	✔	✔	(✔)[5]	✔	?	✔	?
Share	❌	❌	❌	❌	✔	✔	✔	(✔)[5]	✔	?	✔	?
Improve	❌	❌	✓[4]	❌	✔	✔	✔	(✔)[5]	✔	?	✔	?
Moral rights[1]	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	❌	?
Examples	*CompanyX EULA*		*(ZASP)*		*AGPL-3.0* *EUPL-1.2*		*GPL-3.0* *LGPL-3.0* *MPL-2.0* *EPL-2.0* *CC-BY-SA-4.0*		*MIT* *BSD-2-Clause* *BSD-3-Clause* *Apache-2.0* *CC-BY-4.0*		*(CC0-1.0)* *(Unlicense)*	

[1] These are the rights revelant to the “good name” of the author, such as the right to paternity or the right to recall a work. In most jurisdictions in continental Europe they cannot be transferred.

[2] In a EULA the licensee *agrees* to get less rights than what copyright law gives them.

[3] Default situation by law, unless a license is in place.

[4] In some particular cases the licensee has the right to receive or discover the source code and modify it.

[5] For “traditional weak copyleft” the rights to study, share and improve are preserved only for the original library (LGPL) or selected files (MPL, EPL).

🧞 = original/upstream author/licensor
☁ = first user/licensee (= ‘us’), **SaaS provider**
👤 = **SaaS end-user**

Figure 2: FOSS licenses in SaaS environment

FOSS

rights & obligations

FOSS – Permissive / Non-Copyleft

- you receive all rights from your upstream
- the resulting code may be under **any license** (even proprietary)
- you do **not have** to give/offer source code downstream
- include text of license
- keep copyright notices
- sometimes notice of authors

e.g.[3] *BSD-2-Clause*, *BSD-3-Clause*, *MIT*, *Apache-2.0*

[3] The license names in italics follow the SPDX (3.0) nomenclature. The version numbers are referring to the latest version of the license and do not necessarily mean the previous versions do not belong in this category.

FOSS – Copyleft / Share-Alike

- you receive all rights from your upstream
- the resulting code has to be released under the **same license** – give same rights[4]
- you **do have** to give/offer source code downstream
- include text of license
- keep copyright notices (and typically notice of license)
- sometimes notice of changes

e.g. *GPL-3.0-or-later*, *AGPL-3.0-only*, *LGPL-3.0-or-later*, *EPL-2.0*, *MPL-2.0*, *CDDL-1.1*

[4] Copyleft is typically triggered by distribution. In some cases this is narrowed down/weakened (LGPL, MPL, EPL), in others widened/strenghtened (AGPL).

license details

license details

internal use vs (re)distribution

License details – Internal use vs (re)distribution

- **use** = OK by law, no license needed
- copyleft triggered by (re)distribution

https://copyleft.guide

license details

GPL-3.0 vs AGPL-3.0

GPL-3.0 vs AGPL-3.0

in SaaS

copyleft in *AGPL-3.0* triggers also on modification && network use[5]

(L)GPL-* does not

caveat: this works only if the licensee is **using** the software itself (e.g. front-end)

[5] Other SaaS copyleft licenses include *EUPL-1.2* and *OSL-3.0* (and its variations).

license details

LGPL-2.0 vs LGPL-2.1

License details – LGPL-2.0 vs LGPL-2.1

same in spirit

but: *LGPL-**2.1*** introduces the dynamic linking exception

very important for certain languages – e.g. Java

no difference for other languages – e.g. Go

License details – LGPL-2.0 vs LGPL-2.1

LGPL-2.1 §6.b:

> *Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.*

license details

“or later”

License details – “or later” clause

in general

identified in SPDX with: $+$ (e.g. *MPL-1.1+*)

specifics of the GPL family

e.g. *GPL-2.0-or-later* vs *GPL-2.0-only*[6]

- “-only” – 2.0 only
- “-or-later” – 2.0 or any later version (i.e. 2.0, 2.1, 3.0, ...)
- version missing – any version at all (i.e. 1.0, 2.0, 2.1, 3.0, ...)

[6] The “-or-later” and “-only” suffix were (re)introduced for GPL family in SPDX 3 to be more consistent with FSF’s interpretation of their own licenses.

license details

GPL-2.0 vs GPL-3.0

What is new in GPL-3.0

anti-DRM

grace period[7] for violations

explicit patent license

[7] If you need to use GPL-2.0, consider signing https://gplcc.github.io/gplcc/.

license details

the ~~Dalton~~ BSD brothers

BSD-4-Clause (i.e. Averell Dalton)

1. notices in source code
2. notices in binary
3. the “obnoxious BSD advertising clause” – GPL incompatible
4. no endorsement clause

BSD-3-Clause (i.e. Jack Dalton)

1. notices in source code
2. notices in binary
3. ~~the “obnoxious BSD advertising clause” – GPL incompatible~~
4. no endorsement clause

BSD-2-Clause (i.e. William Dalton)

1. notices in source code
2. notices in binary
3. ~~the “obnoxious BSD advertising clause” – GPL incompatible~~
4. ~~no endorsement clause~~

BSD-1-Clause (i.e. Joe Dalton)

1. notices in source code
2. ~~notices in binary~~
3. ~~the “obnoxious BSD advertising clause” – GPL incompatible~~
4. ~~no endorsement clause~~

license details

other licenses in context

Other FOSS licenses in context

permissive licenses (e.g. MIT, ISC) – similar to BSD-3-Clause

- Apache-2.0 – permissive + explicit patent license

Other FOSS licenses in context

permissive licenses (e.g. MIT, ISC) – similar to BSD-3-Clause

- Apache-2.0 – permissive + explicit patent license

weak copyleft differs in scope:

- LGPL-* covers the whole “library”
- *MPL-2.0* covers individual files
- *EPL-1.0* covers the whole “module”, but *EPL-2.0* covers individual files

Other FOSS licenses in context

permissive licenses (e.g. MIT, ISC) – similar to BSD-3-Clause

- Apache-2.0 – permissive + explicit patent license

weak copyleft differs in scope:

- LGPL-* covers the whole “library”
- *MPL-2.0* covers individual files
- *EPL-1.0* covers the whole “module”, but *EPL-2.0* covers individual files

strong copyleft (e.g. GPL-*, AGPL-3.0) covers “the whole derivative work”

Other FOSS licenses in context

permissive licenses (e.g. MIT, ISC) – similar to BSD-3-Clause

- Apache-2.0 – permissive + explicit patent license

weak copyleft differs in scope:

- LGPL-* covers the whole “library”
- *MPL-2.0* covers individual files
- *EPL-1.0* covers the whole “module”, but *EPL-2.0* covers individual files

strong copyleft (e.g. GPL-*, AGPL-3.0) covers “the whole derivative work”

NB: this is a *gross* oversimplification

license details

license compatibility

License compatibility for derived & combined works

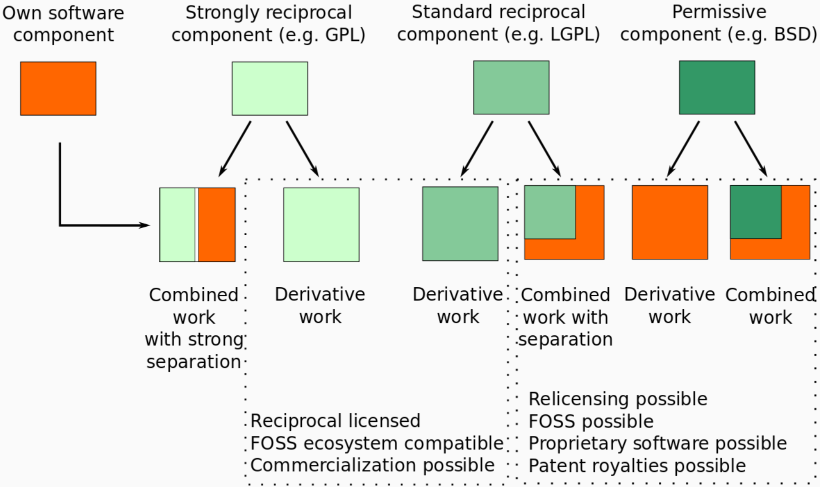

Own software component
Strongly reciprocal component (e.g. GPL)
Standard reciprocal component (e.g. LGPL)
Permissive component (e.g. BSD)
Combined work with strong separation
Derivative work
Derivative work
Combined work with separation
Derivative work
Combined work
Reciprocal licensed
FOSS ecosystem compatible
Commercialization possible
Relicensing possible
FOSS possible
Proprietary software possible
Patent royalties possible

Figure 3: License compatibility for derived & combined works (Mikko Välimäki)

License compatibility between most popular licenses

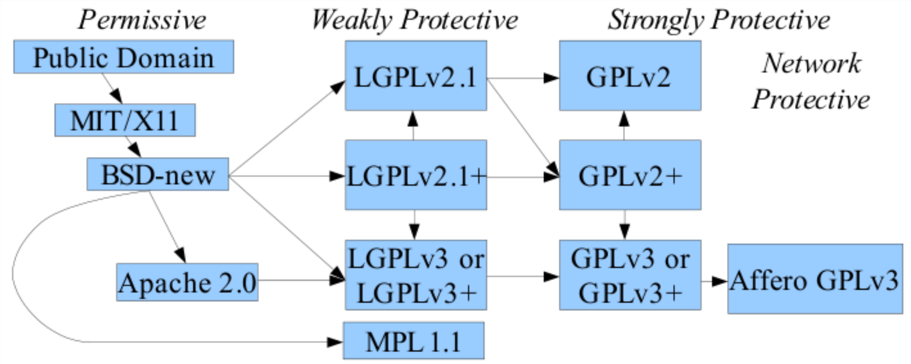
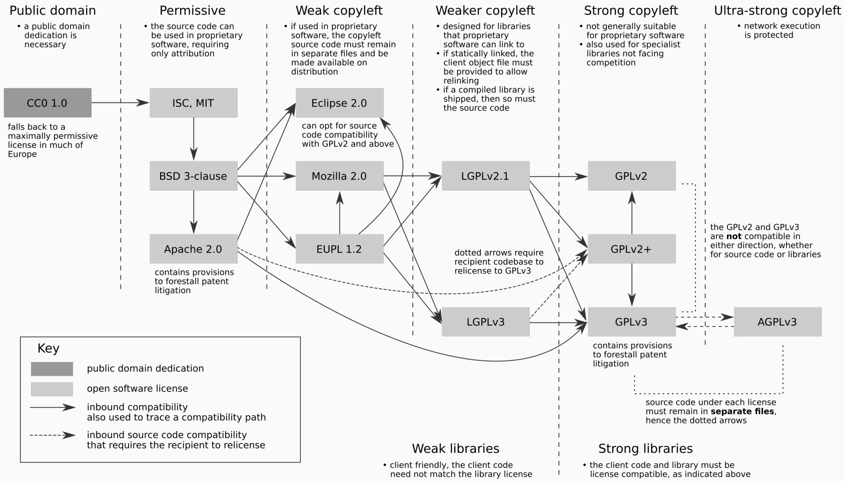


Figure 4: License compatibility between most popular licenses (David A. Wheeler)

License compatibility between most popular licenses



Public domain

- a public domain dedication is necessary

CC0 1.0

falls back to a maximally permissive license in much of Europe

Permissive

- the source code can be used in proprietary software, requiring only attribution

ISC, MIT

BSD 3-clause

Apache 2.0

contains provisions to forestall patent litigation

Weak copyleft

- if used in proprietary software, the copyleft source code must remain in separate files and be made available on distribution

Eclipse 2.0

can opt for source code compatibility with GPLv2 and above

Mozilla 2.0

EUPL 1.2

Weaker copyleft

- designed for libraries that proprietary software can link to
- if statically linked, the client object file must be provided to allow relinking
- if a compiled library is shipped, then so must the source code

LGPLv2.1

LGPLv3

dotted arrows require recipient codebase to relicense to GPLv3

Strong copyleft

- not generally suitable for proprietary software
- also used for specialist libraries not facing competition

GPLv2

GPLv2+

GPLv3

contains provisions to forestall patent litigation

Ultra-strong copyleft

- network execution is protected

the GPLv2 and GPLv3 are **not** compatible in either direction, whether for source code or libraries

AGPLv3

source code under each license must remain in **separate files**, hence the dotted arrows

Key

■ public domain dedication

□ open software license

→ inbound compatibility
also used to trace a compatibility path

⇢ inbound source code compatibility
that requires the recipient to relicense

Weak libraries

- client friendly, the client code need not match the library license

Strong libraries

- the client code and library must be license compatible, as indicated above

Figure 5: License compatibility between most popular licenses (Robbie Morrison)

license details

choose your license

Choose your license: Things to consider

- use case (e.g. traditional distribution vs SaaS)
- compatibility with inbound licenses
- ecosystem (e.g. copyleft is a problem in Go)

Choose your license: Some suggestions

GPL-3.0-or-later – desktop use, you want to be extra sure the software to stay FOSS

AGPL-3.0-or-later, *OSL-3.0*, *EUPL-1.2* – ditto, but in (front-end) SaaS

EPL-2.0, *MPL-2.0*, *LGPL-3.0-or-later* – for libraries, or when you want just your part of code to stay FOSS

Apache-2.0, *BSD-3-Clause*, *MIT* – if wide adoption is key, regardless if in closed source (e.g. standards)

Unlicense, *MIT-0*, *0BSD*, *CC0-1.0*[8] – as "public domain" as it gets

CC-BY-4.0 (permissive), *CC-BY-SA-4.0* (copyleft) – for docs and non-code content

EUPL-1.2 – when you are forced to by public bodies

[8] **NB:** Creative Commons licenses, including *CC0-1.0* **explicitly do not** carry a patent license.

Choose your license: Helper choosers

EC's JoinUp License Assistant

FSF's license recommendation

GitHub's ChooseALicense

marking your own code

marking your own code

how get your software REUSE'd ;)

REUSE: why

why bother?

REUSE: why

why bother?

- no license = no (re)use

REUSE: why

why bother?

- no license = no (re)use
- it is your code – let it be known :)

REUSE: basics

3 easy steps

1. choose and provide licenses
2. add copyright and licensing information to each file
3. (confirm REUSE compliance)

official docs

https://reuse.software/

(much) more on copyright headers

https://matija.suklje.name/how-and-why-to-properly-write-copyright-statements-in-your-code

REUSE example header

```
SPDX-FileCopyrightText: © {$year_of_file_creation} {$name_of_copyright_holder} <{$contact}>
SPDX-License-Identifier: {$SPDX_license_name}
```

REUSE example header

```
SPDX-FileCopyrightText: © {$year_of_file_creation} {$name_of_copyright_holder} <{$contact}>
SPDX-License-Identifier: {$SPDX_license_name}
```

```
SPDX-FileCopyrightText: © 2021 Matija Šuklje <matija@suklje.name>
SPDX-License-Identifier: BSD-3-Clause
```

REUSE example header

```
SPDX-FileCopyrightText: © {$year_of_file_creation} {$name_of_copyright_holder} <{$contact}>
SPDX-License-Identifier: {$SPDX_license_name}
```

```
SPDX-FileCopyrightText: © 2021 Matija Šuklje <matija@suklje.name>
SPDX-License-Identifier: BSD-3-Clause
```

```
SPDX-FileCopyrightText: © 2021 LolWhut Inc. <https://lolwhut.example>
SPDX-License-Identifier: BSD-3-Clause
```

marking your own code

REUSE.software demo

thank you

thank you

15’ break

bonus round

copyright – bonus topics

copyright – bonus topics

moral vs economic

Copyright – Moral & Economic rights

economic rights

rights to economic exploitation of the work

can be transferred (CAA, employment, …)

[9] Technical reason why in jurisdictions with moral © rights you cannot dedicate your work to public domain.

Copyright – Moral & Economic rights

economic rights

rights to economic exploitation of the work

can be transferred (CAA, employment, …)

moral rights

- right to first publication
- right to paternity (= attribution)
- right to integrity (not applicable to code)
- …

cannot be transferred[9]

[9] Technical reason why in jurisdictions with moral © rights you cannot dedicate your work to public domain.

copyright – bonus topics

myths

Copyright – Breaking the myths

- you need to use © sign

Copyright – Breaking the myths

- you need to use © sign
- you need to bump the year in © notice

Copyright – Breaking the myths

- you need to use © sign
- you need to bump the year in © notice
- no license = no copyright

Copyright – Breaking the myths

- you need to use © sign
- you need to bump the year in © notice
- no license = no copyright

lies! damn lies! … and not even statistics

more info: https://matija.suklje.name/how-and-why-to-properly-write-copyright-statements-in-your-code

copyright – bonus topics

exceptions & limitations

Copyright – Exceptions & Limitations

© law allows for some exceptions:

- quotation
- critique
- parody
- school examples
- personal copy
- ...

most of these are of very limited use in writing code (even in research) – do not rely on them

open content & open data

open content & open data

images & design

Copyright in Images & Design

are also covered by copyright[10]

so, you also need a license

[10] Potentially can be covered also under trade dress (similar to trade marks) or industrial design rights (EU) / design patents (US) (needs registration).

open content & open data

free culture

Copyright in Images & Design – Free Culture

- **use** and perform the work
- **study** the work and learn from it
- **share** the original work
- **improve** the work and share your modified work

Copyright in Images & Design – Free Culture licenses

common free culture/content licenses:

- *CC0-1.0* – CC Zero
- *CC-BY-4.0* – CC Attribution
- *CC-BY-SA-4.0* – CC Attribution ShareAlike
- *OFL-1.1* – SIL Open Font License
- *CERN-OHL-1.2* – CERN Open Hardware License

problematic licenses / false friends:

- *CC-BY-SA-NC-4.0* – CC [...] **NonCommercial**
- *CC-BY-SA-ND-4.0* – CC [...] **NoDerivatives**

open content & open data

open data

Data basics

information is not © protected *per se*

sets of data and data bases are protected by *sui generis* database rights[11]

[11] In the EU. In USA and some other jurisdictions, data bases fall under (a “lesser”) copyright.

Open Data licenses

common open data licenses:

- *CC0-1.0* – CC Zero
- *PDDL-1.0* – ODC[12] Public Domain Dedication & License
- *ODC-By-1.0* – Open Data Commons Attribution License
- *ODbL-1.0* – ODC Open Database License (ShareAlike)

[12] Open Data Commons

FOSS Licensing

Part 2 – Inbound compliance & tooling

Matija Šuklje

September 2022

University of Maribor, Open Science Summer School

plan

plan

what we will look into today

Agenda

Part 2: inbound licensing

1. why compliance matters
2. OpenChain – compliance ISO standard
3. SPDX – data ISO standard
4. tools & services
5. tools demo

FOSS compliance

FOSS compliance

why bother?

FOSS compliance: What is it?

= bare *minimum* legal requirement

failure to follow the licenses can result in copyright infringement

FOSS compliance: What is it?

= bare *minimum* legal requirement

failure to follow the licenses can result in copyright infringement

civil & criminal repercussions possible

FOSS compliance

what to watch out for

FOSS: In practice

inbound licenses need to be compatible with your outbound licenses

you have to:

- keep license texts & copyright notices
- properly mark your own work
- if copyleft, resulting code has to be released under the same license[1]
- if copyleft, give/offer source code & log if you changed upstream code[2]
- (look for licenses that your downstream is also OK with)
- (list all 3rd party code and info about it)

[1] Copyleft is typically triggered by distribution.

[2] Necessary just for some licenses, but always a good idea!

FOSS compliance: Technical questions

who holds the (copy)rights?

what are the applicable licenses?

is the code (un)modified?

how do the differently licensed pieces of code interact?

FOSS compliance: Technical questions

who holds the (copy)rights?

what are the applicable licenses?

is the code (un)modified?

how do the differently licensed pieces of code interact?

no derivative work

- runtime (e.g. CLI)
- web API (e.g. REST)

(probably) derivative work

- linking (static vs dynamic – important for e.g. *LGPL-2.1-or-later*)
- copy-paste

standards

standards

OpenChain

OpenChain

FOSS compliance minimum standards[3] and certification

- people
- policies
- processes

(self-certification possible)

https://openchainproject.org/

[3] ISO standard ISO/IEC 5230:2020.

standards

SPDX

SPDX = Software Package Data Exchange

standard[4] for communicating software bill of material information on, including:

- components,
- licenses, copyright,
- security references,
- code relationships, etc. technical details

also license unique IDs, language/syntax and file formats for all of above

https://spdx.dev/

[4] ISO standard ISO/IEC DIS 5962.

SPDX: Words and syntax

SPDX unique license names, e.g.:

BSD 3-Clause “New” or “Revised” License = BSD-3-Clause

https://spdx.org/licenses/

SPDX: Words and syntax

SPDX unique license names, e.g.:

BSD 3-Clause “New” or “Revised” License = BSD-3-Clause

https://spdx.org/licenses/

simple SPDX IDs:

```
SPDX-License-Identifier: LGPL-2.1-or-later
```

[5] SPDX uses the following operands: `AND`, `OR`, `WITH`, `+`, and parantheses.

SPDX: Words and syntax

SPDX unique license names, e.g.:

BSD 3-Clause "New" or "Revised" License = BSD-3-Clause

https://spdx.org/licenses/

simple SPDX IDs:

```
SPDX-License-Identifier: LGPL-2.1-or-later
```

complex SPDX expressions[5]:

```
SPDX-License-Identifier: Apache-2.0 AND (MIT OR GPL-2.0-only)
```

https://spdx.dev/ids/

[5] SPDX uses the following operands: `AND`, `OR`, `WITH`, `+`, and parantheses.

SPDX: File[6]

important SPDX tags:

- `PackageCopyrightText` – copyright notice in package
- `PackageLicenseDeclared` – license ID the package claims to be under
- `PackageLicenseInfoFromFiles` – license ID that tool found in the package
- `PackageLicenseConcluded` – license ID a human concluded as actual state of package
- `FileCopyrightText` – copyright notice in file
- `LicenseInfoInFile` – license ID that tool found in the file
- `LicenseConcluded` – license ID a human concluded as actual state of file

[6] SPDX supports several formats: tag:value, RDF, JSON, YAML.

standards

purl

package URL

URL to identify packages and their origin

```
scheme:type/namespace/name@version?qualifiers#subpath
```

e.g.:

```
pkg:github/biolab/orange3@220d4bc543369c6735ff939c2cfb8e43da595327
pkg:npm/foobar@12.3.1
pkg:golang/google.golang.org/genproto#googleapis/api/annotations
```

is also used in SPDX

https://github.com/package-url/purl-spec

useful tools and services

useful tools and services

ClearlyDefined.io

ClearlyDefined.io

license & copyright data (& collaboration)

offers a confidence score (e.g. 87% is really high)

web service & REST API

https://clearlydefined.io/

useful tools and services

ScanCode

NexB ScanCode

license & copyright scanner

scans only the code you fed it

one-shot workflow (well suited for CI/CD[7])

CLI

https://github.com/nexB/scancode-toolkit

has a pipeline-based server: https://scancode.io

[7] See e.g. Oracle's License File Auditor as its integration into GitHub Actions.

useful tools and services

ORT

OSS Review Toolkit

- Analyzer - determines the dependencies and their meta-data
- Downloader - fetches all source code of the projects and their dependencies
- Scanner - uses configured source code scanners (ScanCode by default)
- Advisor - retrieves security advisories for used dependencies
- Evaluator - evaluates license / copyright info against customizable policy rules
- Reporter - presents results in various formats

emphasis on CI/CD

CLI

https://oss-review-toolkit.org

useful tools and services

OpossumUI

OpossumUI

auditing/reviewing tool

consumes and integrates data from ORT, ScanCode, OWASP, ScanOSS, SPDX[8]

desktop/WebUI

https://github.com/opossum-tool/OpossumUI

[8] SPDX 2.2 in JSON or YAML format.

useful tools and services

FOSSology

FOSSology

license & copyright (& export control, patent) scanning suite

scans only the code you fed it (+ unpacks any archives within)

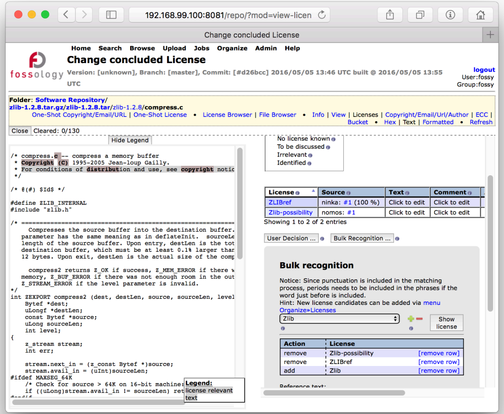
auditing workflow

integrates several scanning agents

WebUI + REST API

https://fossology.org

FOSSology


192.168.99.100:8081/repo/?mod=view-licen
Change concluded License
Home Search Browse Upload Jobs Organize Admin Help
fossology
logout
One-Shot Copyright/Email/URL | One-Shot License • License Browser | File Browser • Info | View | Licenses | Copyright/Email/Url/Author | ECC | Bucket • Hex | Text | Formatted • Refresh
Close
Hide Legend
No license known
To be discussed
Irrelevant
Identified
User Decision ...
Bulk Recognition ...
Show license

Change concluded License

Version: [unknown], Branch: [master], Commit: [#d26bcc] 2016/05/05 13:46 UTC built @ 2016/05/05 13:55 UTC

User:fossy
Group:fossy

Folder: Software Repository/zlib-1.2.8.tar.gz/zlib-1.2.8.tar/zlib-1.2.8/**compress.c**

Cleared: 0/130

```
/* compress.c -- compress a memory buffer
 * Copyright (C) 1995-2005 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notic
 */

/* @(#) $Id$ */

#define ZLIB_INTERNAL
#include "zlib.h"

/* ===========================================================
     Compresses the source buffer into the destination buffer.
   parameter has the same meaning as in deflateInit.  sourceLe
   length of the source buffer. Upon entry, destLen is the tot
   destination buffer, which must be at least 0.1% larger thar
   12 bytes. Upon exit, destLen is the actual size of the comp

     compress2 returns Z_OK if success, Z_MEM_ERROR if there w
   memory, Z_BUF_ERROR if there was not enough room in the out
   Z_STREAM_ERROR if the level parameter is invalid.
*/
int ZEXPORT compress2 (dest, destLen, source, sourceLen, level
    Bytef *dest;
    uLongf *destLen;
    const Bytef *source;
    uLong sourceLen;
    int level;
{
    z_stream stream;
    int err;

    stream.next_in = (z_const Bytef *)source;
    stream.avail_in = (uInt)sourceLen;
#ifdef MAXSEG_64K
    /* Check for source > 64K on 16-bit machine:
    if ((uLong)stream.avail_in != sourceLen) ret
```

Legend:
license relevant text

License	Source	Text	Comment
ZLIBref	ninka: #1 (100 %)	Click to edit	Click to edit
Zlib-possibility	nomos: #1	Click to edit	Click to edit

Showing 1 to 2 of 2 entries

Bulk recognition

Notice: Since punctuation is included in the matching process, periods needs to be included in the phrases if the word just before is included.
Hint: New license candidates can be added via menu Organize»Licenses

Zlib

Action	License	
remove	Zlib-possibility	[remove row]
remove	ZLIBref	[remove row]
add	Zlib	[remove row]

Reference text:

Figure 1: FOSSology: License view of a file with bulk recognition function

useful tools and services

SW360

Eclipse SW360

software catalogue manager

suited for complex projects/products or when they share components

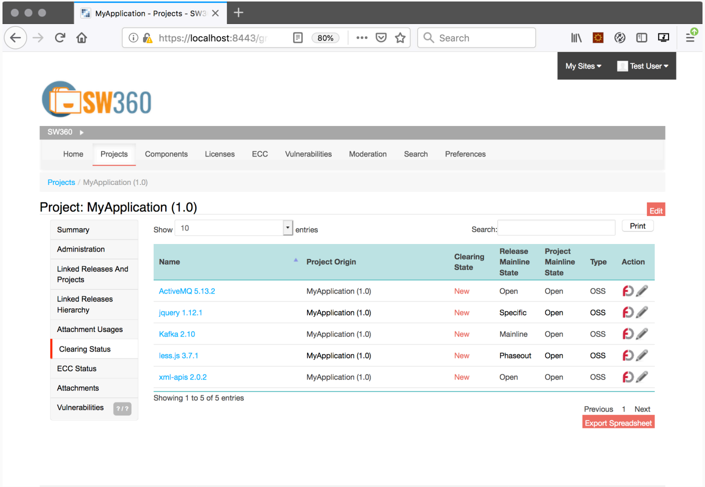
integrates with many scanners and other tools (e.g. FOSSology)

(SW360Antenna is its optional automation component)

WebUI + REST API

https://www.eclipse.org/sw360/

Eclipse SW360



Project: MyApplication (1.0)

Name	Project Origin	Clearing State	Release Mainline State	Project Mainline State	Type	Action
ActiveMQ 5.13.2	MyApplication (1.0)	New	Open	Open	OSS	
jquery 1.12.1	MyApplication (1.0)	New	Specific	Open	OSS	
Kafka 2.10	MyApplication (1.0)	New	Mainline	Open	OSS	
less.js 3.7.1	MyApplication (1.0)	New	Phaseout	Open	OSS	
xml-apis 2.0.2	MyApplication (1.0)	New	Open	Open	OSS	

Showing 1 to 5 of 5 entries

Figure 2: SW360: Project view with (license) clearing information shown

useful tools and services

(more) specialised tools

Extra specialised tools

- SPDX Tools – to handle SPDX files
- Tern – for Docker images
- BANG – for firmware images
- CLA Assistant – **if** you need people to sign a CLA for your project[9]

more info on:

http://oss-compliance-tooling.com/

[9] If you **really** need a CLA, please look at FLA-2.0 on https://contributoragreements.org/ca-cla-chooser/. Also my blog post on it.

tooling demo

tooling demo

ScanCode & SPDX & OpossumUI

ScanCode & SPDX demo backup recording

https://asciinema.org/a/ZJk3rrmymbVonySILxwxCpKF7

or in terminal:

```
asciinema play https://asciinema.org/a/ZJk3rrmymbVonySILxwxCpKF7
```

FOSS governance

FOSS governance

more than compliance

FOSS governance: The next step

= consistent *policies*, *processes*, and *decisions* regarding FOSS

- beyond mere compliance
- interact with the FOSS community
- contribute back to upstream
- collaboration reduces R&D costs in the long run
  FOSS = critical supplier *… treat it as such*

the end

the end

thank you

Questions & contact

Any questions remaning?

Questions & contact

Any questions remaning?

Matija Šuklje

matija@suklje.name

https://matija.suklje.name