IN5320

Free and Open Source Software

University of Oslo
Department of Informatics

Outline

- What is FOSS?
- FOSS as an approach to software development
- Business models for FOSS development
- Intellectual property rights - copyrights and patents
- Software licenses
- FOSS and Platform Ecosystems

Free and Open Source Software

[…] anyone is freely licensed to use, copy, study, and change the software in any way, and the source code is openly shared so that people are encouraged to voluntarily improve the design of the software.

– Wikipedia

Types of software

Software type	Free (cost)	Redistri-butable	Unlimited use and users	Source code available	Source code modifiable
Commercial	-				
Shareware	X	X			
Freeware	X	X	X		
Royalty-free libraries	X	X	X	X	
Open source	X	X	X	X	X

Historical context

- In the early days of programming, sharing of software among programmers was the norm
- Hardware vendors started to dominate software distribution in the 1980s, releasing proprietary software in binary form
- FSF established in 1985 to re-establish free software norms - defined **4 freedoms for software**
- In the second half of the 1990s, the internet facilitated distributed OSS development
- Open Source Initiative (OSI) founded in 1998 to promote OSS as a solution for businesses - defined **10 criteria for OSS**

1. Free redistribution

- No restriction on redistribution (free or paid) of the software

- The software can be redistributed alone or as a component of an aggregate software distribution

- The licensee can not require a royalty or fee for redistribution

2. Source code

- The software must include the source code, or it must be easily obtainable through well-published means

- The source code should not be deliberately obfuscated, and intermediate forms (preprocessor/translator output) are not allowed.

3. Derived works

- The license must allow modification and derived works

- Derived works must be allowed to be distributed under the same terms as the original

4. Integrity of the author's source code

- The license may only restrict source-code from being distributed in modified form if it allows patch-files that can modify it at build time

- The license might require derived works to use a different name and/or version number

5. No discrimination against persons or groups

- No discrimination against persons or groups

6. No discrimination against fields of endeavour

- Restrictions in the use of the software in particular fields of endeavour is not allowed

7. Distribution of license

- License for software must also apply to those it is redistributed to

8. License must not be specific to a product

- License for software must not depend on it being part of a software distribution
- The same license must apply if the software is extracted from a distribution and distributed separately

9. License must not restrict other software

- The license must not place restrictions on other software that is distributed alongside the licensed software

10. License must be technology-neutral

- No provision of the license may be predicated on any individual technology or style of interface

Four freedoms for software

0. The freedom to run the program, for any purpose.

1. The freedom to study how the program works, and change it to make it do what you wish.

2. The freedom to redistribute copies so you can help your neighbour.

3. The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes.

Source: https://www.gnu.org/philosophy/free-sw.en.html

Free vs Open

- Free software refers to freedom, not cost - "free speech", not "free beer"
- Based on promoting social solidarity and sharing
- *Free* software meet the 10 criteria for open source
- Practical difference: *free* licenses (e.g. GPL) require derivative work to be open source to ensure that software remains free - more later

Models for production of software

1. Managerial command systems - firms and organisations with "lines of command"; centralised hierarchy in which tasks are defined and distributed

2. Markets - transaction costs define the production; tasks are tagged with a price to attract workers

3. Commons Based Peer Production

- FOSS can follow any of the models, but peer production is the "typical" example

Commons Based Peer Production

- A "[…] model of socioeconomic production in which large numbers of people work cooperatively (usually over the Internet)" (Wikipedia).
- Coined by Benkler (2002), in a study of open source software development
- No clear-cut distinction with crowdsourcing, but usually involves a stronger sense of community

Conditions for CBPP

- Premise: existence of excess capacity, resulting from a great number of potential contributors and a set of organisational structures
- CBPP model requires that work can be modularised
  - Work is divided into modules which can be:
    - independently and incrementally produced
    - sufficiently fine-grained to allow the capture of small contributions
    - quality-checked and integrated with the overall system through reasonably low cost mechanisms

Examples of CBPP

- Free and Open Source Software was the inspiration for the CBPP project
- Wikipedia - ± 30 000 active contributors with 5+ edits per month
- OpenStreetMaps - ±50 000 active contributors per month

FOSS development approach

- Decentralised geographically - internet key infrastructure
- Rapid evolution with frequent, incremental releases
- Real-world meetings are seldom, coordination happens in various online tools
- Version control and source code repositories critical (GitHub, sourceforge etc)
- Dominated by operating and networking system software, development tools and infrastructural component, for example linux, apache web server, python, V8 JS engine, react, angular etc.

FOSS 2.0

- Fitzgerald (2006): open source is transforming from its "free software" origins to a more mainstream and commercially viable approach - FOSS 2.0
- Classic (early) example:
    - One single or a small group of developers establishes a project and its direction
    - Other developers submit patches to fix bugs or add functionality
    - Examples: apache web server, fetchmail, emacs
- Increasingly (OSS 2.0):
    - Companies establish OSS projects as part of a purposeful strategy
    - Developers are paid to contribute
    - Examples: React and Angular largely developed by Facebook and Google; Linux kernel top 10 contributors include Intel, Red Hat, Samsung, IBM

FOSS participants

- Key stakeholders or participants in FOSS development:
  - Individual developers - often perceived as "hobbyists", but in reality often full-time developers
  - Companies supporting development and distribution
  - Users - experts and early adopters, often the same people who contribute to open source projects
- Motivation for participation in FOSS projects:
  - Technical and economic
  - Socio-technical

Technical and economical motivation

- OSS seen as having potential to address "Software crisis" - software taking too long to develop, not working well when delivered, and costing too much

  - Speed - OSS characterised by short development cycles. "Adding manpower to a late software project makes it later" vs "given enough eye-balls, every bug looks shallow".

  - Quality - peer review of source code. Some argue OSS devs are among the most talented and motivated.

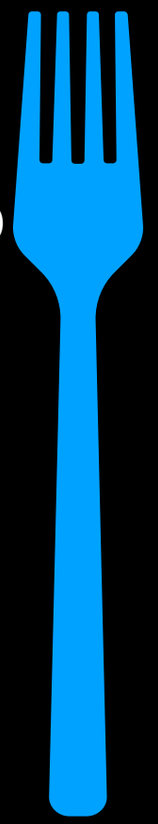
  - Cost - shared costs and shared risks of development.

Socio-technical motivation

- Motivation of individual developers often socio-technical
- Studies point to "rush" of being able to produce something that get feedback and is used by others
- Meritocracy, where quality of code speaks for itself
- Arena for demonstrating skills for potential employers
- Different in OSS projects where developers are paid

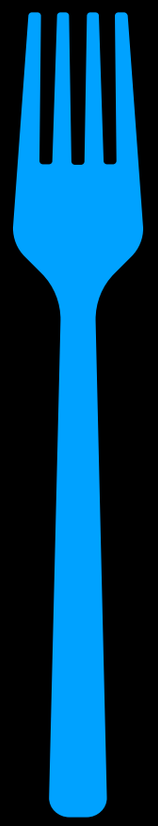
Forks

- Often no written rules within open source projects - customs and taboos must be learned by experience
- FOSS repositories are not open for anyone to commit to without approval
- The right to **fork** is central to FOSS - defined as a new independent line of development, by making a copy of the source code which is then developed separately
- However, forking is often seen as bad practice



Forks

- Examples of well-known forks:
  - OpenOffice => LibreOffice
  - KHTML => WebKit => Blink
  - Mambo => Joomla
  - Debian => Ubunutu
  - Android => Fire OS++
  - Visualisation of linux forks



Business models

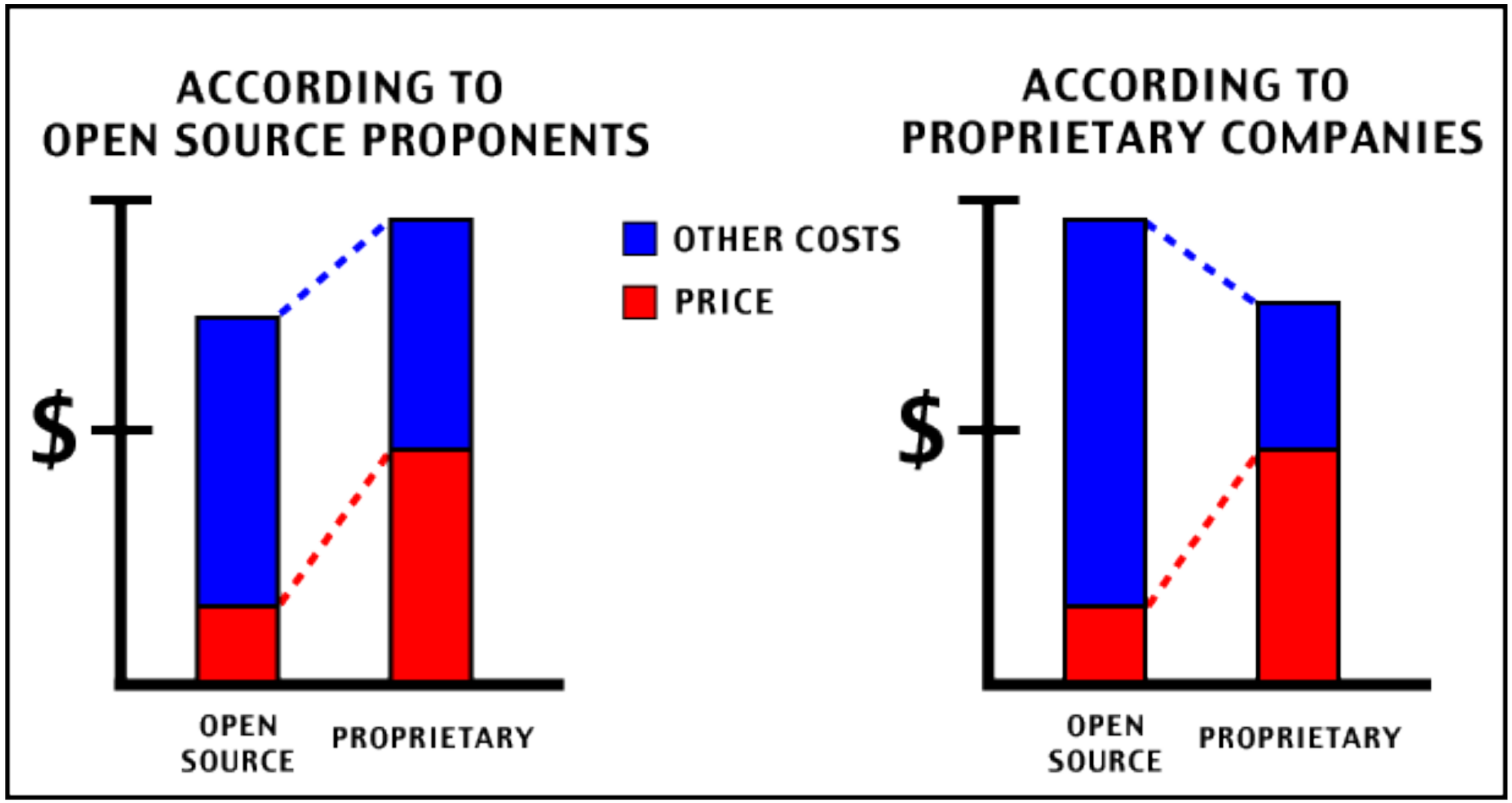
- Business model: how an organisation creates value
- Major organisations base their business on OSS - Red Hat, SUSE, Canonical, Apache Foundation, Mozilla, eZ System
- Other organisations use OSS without having it as a main business - IBM, Google, Apple, Oracle

Cost Reduction

- OSS *can* help reduce cost
- Depends on TCO of OSS vs the alternatives
- Applicable when software sale is not the main revenue

- Example: Sun Microsystems buying the company behind what would become OpenOffice, to reduce licensing cost (and market share) of MS Office. LibreOffice was forked from OpenOffice.org in 2010

Cost Reduction



Source: Northwest Regional Educational Laboratory, Portland, Oregon

Services

- Offering services based on OSS: web hosting, file hosting, infrastructure/platform/software as a service (IaaS/PaaS/SaaS)

- Often combined with the freemium model

> *It's obvious, if we don't support Linux, we'll be Windows only and that's not practical.*
>
> Mark Russinovich, CTO of Microsoft Azure

- Example: Linode (or other VSP) who provide hosting of servers running different versions of linux

Support and consulting

- Charging for consulting, support and maintenance of OSS
- Configuration of complex software, providing training etc

- Example: Canonical, who develops the Ubuntu linux distribution and makes money from support and consulting related to it.

Loss-leader

- Providing a product for free or low cost to increase the market and/or attract sales of related products
- Often combined with dual-licensed software
- Examples:
  - IBM open sourcing Eclipse IDE, in order to increase market for related products.
  - MySQL providing open source community edition to drive sales of commercial edition

Other

- Freemium - providing a free basic tier to attract users
- Open core - open sourcing the core product, but with certain parts under a proprietary license
- Hardware - using OSS in hardware products such as routers, TVs etc
- Accessorising - selling accessories related to OSS
- Advertising and search - ads and search engines
- Donations

Process	FOSS	OSS 2.0

Development Life Cycle	• Planning—"an itch worth scratching"
	• Analysis—part of conventional agreed-upon knowledge in software development
	• Design—firmly based on principles of modularity to accomplish separation of concerns
	• Implementation
	  ○ Code
	  ○ Review
	  ○ Pre-commit test
	  ○ Development release
	  ○ Parallel Debugging
	  ○ Production Release
	(often the planning, analysis, and design phases are done by one person/core group who serve as "a tail-light to follow" in the bazaar)
		• Planning—purposive strategies by major players trying to gain competitive advantage
		• Analysis and design—more complex in spread to vertical domains where business require-ments not universally understood
		• Implementation subphases as with FOSS, but the overall development process becomes *less* bazaar-like
		• Increasingly, developers being paid to work on open source

Product Domains	• Horizontal infrastructure (operating systems, utilities, compilers, DBMS, web and print servers)	• More visible IS applications in vertical domains

Primary Business Strategies	• Value-added service-enabling
	• Loss-leader/market-creating
		• Value-added service enabling
		  ○ Bootstrapping
		• Market-creating
		  ○ Loss-leader
		  ○ Dual product/licensing
		  ○ Cost reduction
		  ○ Accessorizing
		• Leveraging community development
		• Leveraging the open source brand

Product Support	• Fairly haphazard—much reliance on e-mail lists/bulletin boards, or on support provided by specialized software firms	• Customers willing to pay for a professional, whole-product approach

Fitzgerald 2006

Intellectual Property

- Tangible assets:
    - properties, currencies, equipment…
- Intangible assets:
    - knowledge, experience, (social) networks, brand loyalty…
    - more formalised: copyrights, patents, trademarks…
- Intellectual Property Rights are rights to *intangible assets*

Intellectual Property

- Intellectual property:
  - "Non-physical property that is the product of original thought".

    Stanford Encyclopaedia of Philosophy
  - "[IP] refers to creations of the intellect for which a monopoly is assigned to designated owners by law".

    Wikipedia

- Intellectual property **rights** do not address the abstract idea, but the physical manifestation or expression of ideas

- Covered by international treaties (e.g. Bern convention from 1886) and national law in most of the world, but laws differ

Why IPR?

- Three (philosophical) arguments for IPR:
  - Intellectual property is an expressions of ones personality, thus it should be possible to protect
  - People have the rights to the results of their labour, also for intellectual property rights
  - Granting rights of ownership necessary to incentivise creation of intellectual works

IPR protection

- Protection of IPR is mainly through:
  - Copyrights
  - Patents
  - Trade secrets
  - Trademarks

Disclaimer

- Intellectual property rights, copyright, patents, licensing is complicated and food for lawyers
- If doing this for real it can be a good idea to involve lawyers to understand all the details

- The goal of this lecture is to give an understanding of the domain and some of the challenges

Copyrights and Patents

- Patents and copyrights are the main instruments of IPR law
- History and purpose are different:
    - Patents are issued by authorities to regulate use of **inventions and ideas** for commercial uses
    - Copyrights applies to the **expression of works** like printed material, sound recordings, software - not ideas

Copyright

- Protection of original works of authorship "fixed in any medium of expression"
- Can be applied to literature, music, photography, architecture, maps, **software etc**
- Must be original, produced by the author
- Must be "non-utilitarian" and "non-functional"
- Do not cover abstract ideas themselves

Copyright

- Copyright laws address *use* of material - not a means to control access
- Use of content includes:
  - distribution of unaltered content
  - distribution of content in a collection
  - distribution of adaptions and derivate work
  - performing and distributing produced work

Rights of copyright holders

- Copyright owners can:
  - reproduce the work
  - adapt or derive other works from the original work
  - distribute copies of the work
  - display the work publicly
  - perform the work publicly

Copyright limitations

- Copyright is time bound - normally a number of years (70) after death of author

- Two general limitations:
  - Fair use - limited use of copyrighted work is allowed, for commenting, news reporting, research, teaching etc
  - First sale - copyright holders who have sold copies of a work cannot interfere with subsequent sales of those copies

Patents

- Concrete solution to a practical problem - processes, products, medicines, applications
- Must be applied for to national patent authorities, and specifics varies by country/legislation
- Types of patents include *inventions* and *design*
- Requirements
  - Useful
  - Novel
  - Non-obvious

Patent holder rights

- Patent holders can:
  - make the patented item
  - use the patented item
  - sell the patented item
  - authorise others to sell the patented item
- No-one can patent or market the same process or item while the patent is valid, even if it was invented independently
- Patents are time bound - normally 20 years

Software patents

- Patentability of software is disputed, and varies across the world:
  - Europe: "programs for computers" are excluded
  - US: software is patentable
  - In general: *limitations* on software patents are common
- Many are critical of software patents and its growth

Balancing IPR

protecting rights of author/inventor to incentivise creation

making works and inventions available to the benefit of the public

- IPR law aims to strike balance between incentivising creators and making sure society benefits from creations
- IP protection has expanded over time
- First copyright law: 14 years from work was created - current (US) law: up to 120 years

Public Domain

- Works for which IPR have expired or been waived are in the **public domain**

- Works in the public domain are free of any restrictions and can be used by anybody in any way

Licensing

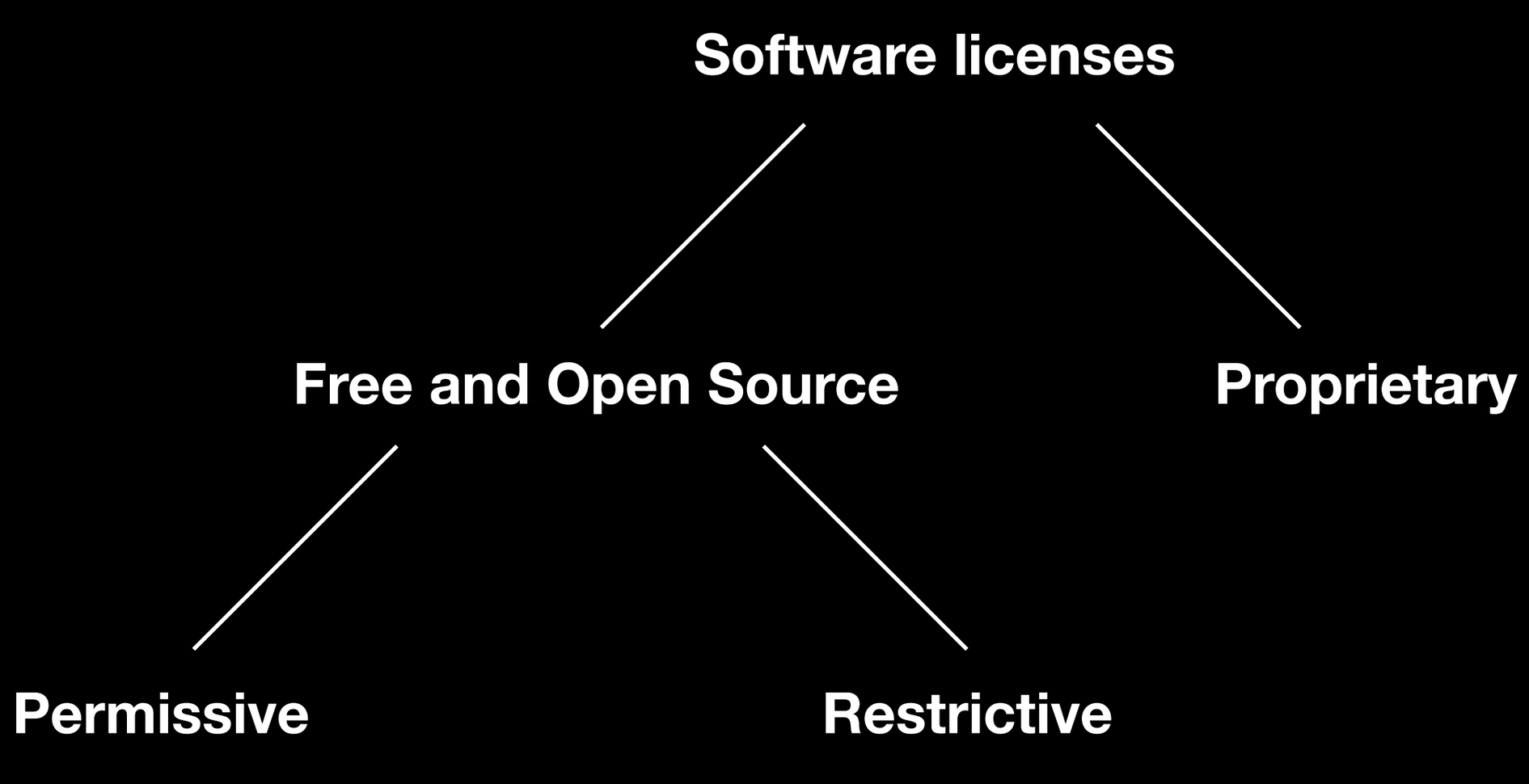
- IPR grant rights to authors of their work - including authors of software

- Providing content without license information is legal, but can create confusion

- To use intellectual property written by someone else a *license* is required - including for software

Software Licenses

- Software is created by an author and is subject to *copyright*
- A *license* is needed for software to be used by others

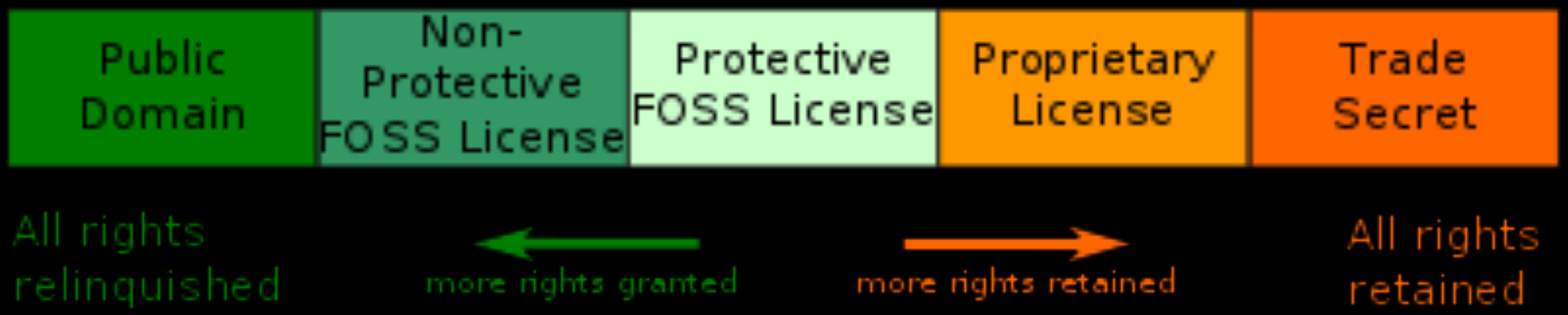


Restrictive vs Permissive

- Permissive licenses *allow* distribution of source code, but only *require* attribution - "minimal restrictions on future behaviour" (FreeBSD)

- Restrictive (copyleft) licenses *require* source code to be distributed along with binary code - aim to keep software free in the future

Software licenses



Source: Mark Webbink

Copyleft - all rights reversed

- Restrictive vs permissive goes back to philosophical differences between *free* and *open*

GNU is not in the public domain. Everyone will be permitted to modify and redistribute GNU, but *no distributor will be allowed to restrict its further redistribution*. That is to say, proprietary modifications will not be allowed. I want to make sure that all versions of GNU remain free.

– Richard Stallman

Restrictive licenses

- Weak restrictive/copyleft license:
    - If software with weak copyleft is used *that module/ library's source code must be distributed/made available*
- Strong restrictive/copyleft license:
    - If software with strong copyleft is used *the entire software's source code must be distributed/made available*

Viral licenses

- Strong copyleft is *viral*
- When used they force the entire application to be released under strong copyleft license

Example: MIT

- Example of permissive license
- Grants linsence permission to use the software in any way, with only one condition:

  The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

Example: GNU GPL

- General Public License - primary example of copyleft
- Several versions:
  - GPL (currently v3)
  - Lesser GPL - weak copyleft license
  - Affero General Public License (AGPL)

Example: Affero GPL

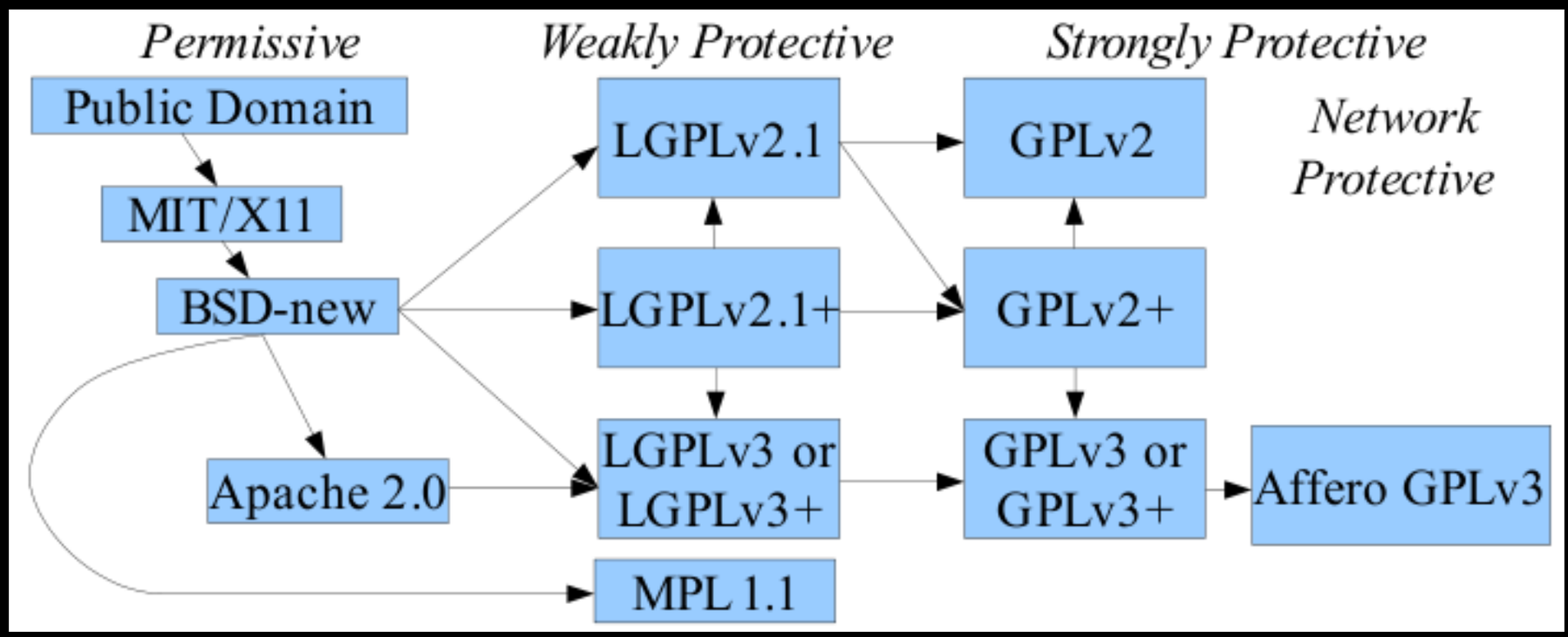
- Designed to address perceived loophole in the GPL, in cases where GPL-licensed software is used to provide cloud services

- Based on GPL, but with an added provision that address *use* of software over a *network*

- Requires source code to be made available to *users* that access the software over a network

License compatibility

- Not all licenses are compatible
- Compatible here means that source code under one license can be part of software distributed under another license



		I want to license my code under:					
		GPLv2 only	**GPLv2 or later**	**GPLv3 or later**	**LGPLv2.1 only**	**LGPLv2.1 or later**	**LGPLv3 or later**
I want to copy code under:	**GPLv2 only**	OK	OK [2]	NO	OK: Combination is under GPLv2 only [7]	OK: Combination is under GPLv2 only [7][2]	NO
	GPLv2 or later	OK [1]	OK	OK	OK: Combination is under GPLv2 or later [7]	OK: Combination is under GPLv2 or later [7]	OK: Combination is under GPLv3 [8]
	GPLv3	NO	OK: Combination is under GPLv3 [3]	OK	OK: Combination is under GPLv3 [7]	OK: Combination is under GPLv3 [7]	OK: Combination is under GPLv3 [8]
	LGPLv2.1 only	OK: Convey copied code under GPLv2 [7]	OK: Convey copied code under GPLv2 or later [7]	OK: Convey copied code under GPLv3 [7]	OK	OK [6]	OK: Convey copied code under GPLv3 [7][8]
	LGPLv2.1 or later	OK: Convey copied code under GPLv2 [7][1]	OK: Convey copied code under GPLv2 or later [7]	OK: Convey code under GPLv3 [7]	OK [5]	OK	OK
	LGPLv3	NO	OK: Combination is under GPLv3 [8][3]	OK: Combination is under GPLv3 [8]	OK: Combination is under GPLv3 [7][8]	OK: Combination is under LGPLv3 [4]	OK
I want to use a library under:	**GPLv2 only**	OK	OK [2]	NO	OK: Combination is under GPLv2 only [7]	OK: Combination is under GPLv2 only [7][2]	NO
	GPLv2 or later	OK [1]	OK	OK	OK: Combination is under GPLv2 or later [7]	OK: Combination is under GPLv2 or later [7]	OK: Combination is under GPLv3 [8]
	GPLv3	NO	OK: Combination is under GPLv3 [3]	OK	OK: Combination is under GPLv3 [7]	OK: Combination is under GPLv3 [7]	OK: Combination is under GPLv3 [8]
	LGPLv2.1 only	OK	OK	OK	OK	OK	OK
	LGPLv2.1 or later	OK	OK	OK	OK	OK	OK
	LGPLv3	NO	OK: Combination is under GPLv3 [9]	OK	OK	OK	OK

Source and object code

- Distinction relevant for licensing:
  - source code - human/programmer-readable and editable software
  - object code - compiled, binary software
- Some languages are never distributed in compiled form

Distributing OSS

- Requirement to distribute source code in open source licenses is linked to distribution of object/binary code

- *Internal* modification and use of OSS software does not usually trigger requirement to publish modified code

- Businesses may (should) have a list of accepted OSS licenses and used OSS modules - example

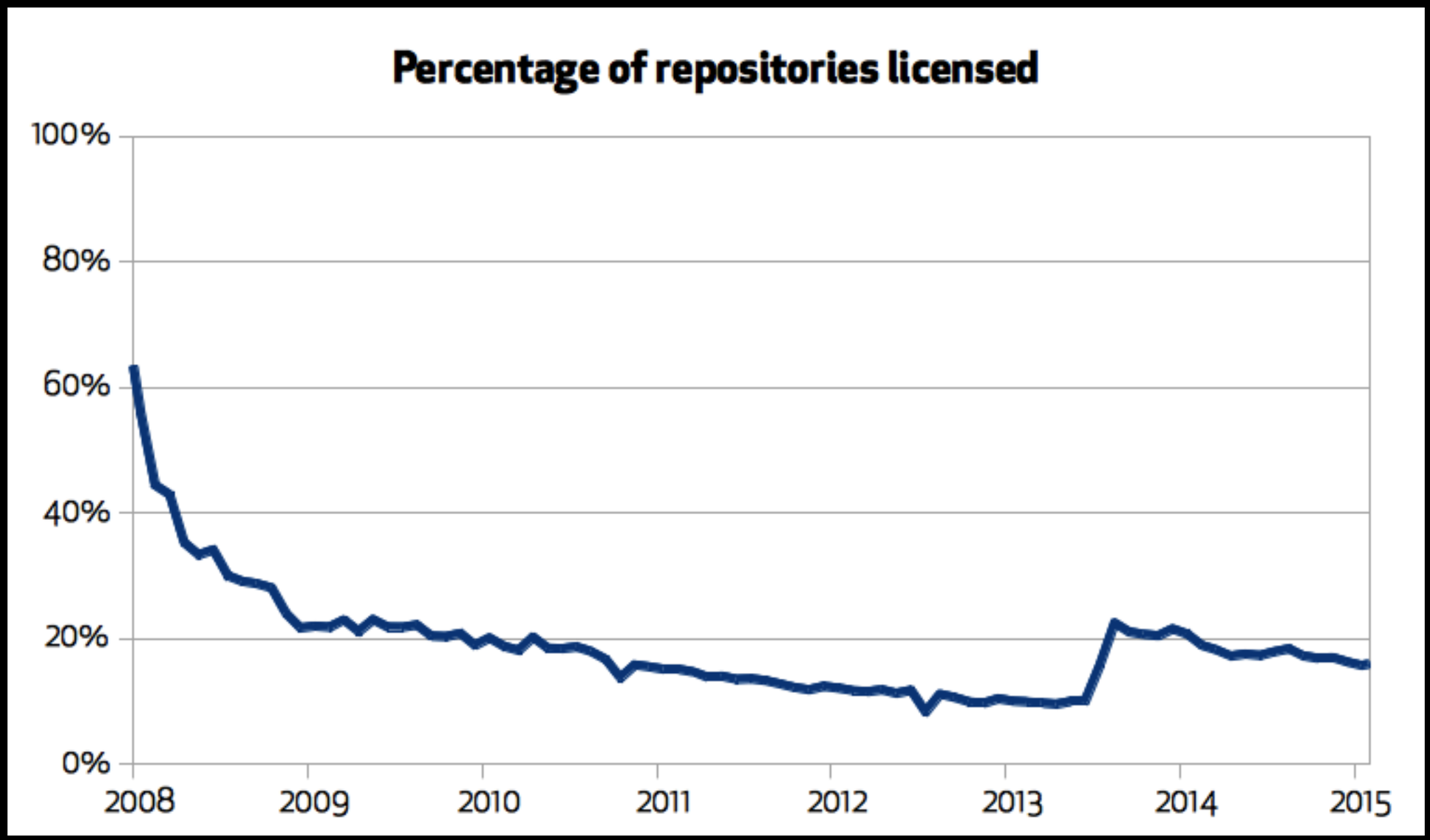
Distributing OSS

- Choice of license is important for the open source project - affects economic and growth potential

- https://choosealicense.com

- Topic of research, e.g. Hoffman et al (2013) - correlate choice of license with growth of project

Distributing OSS



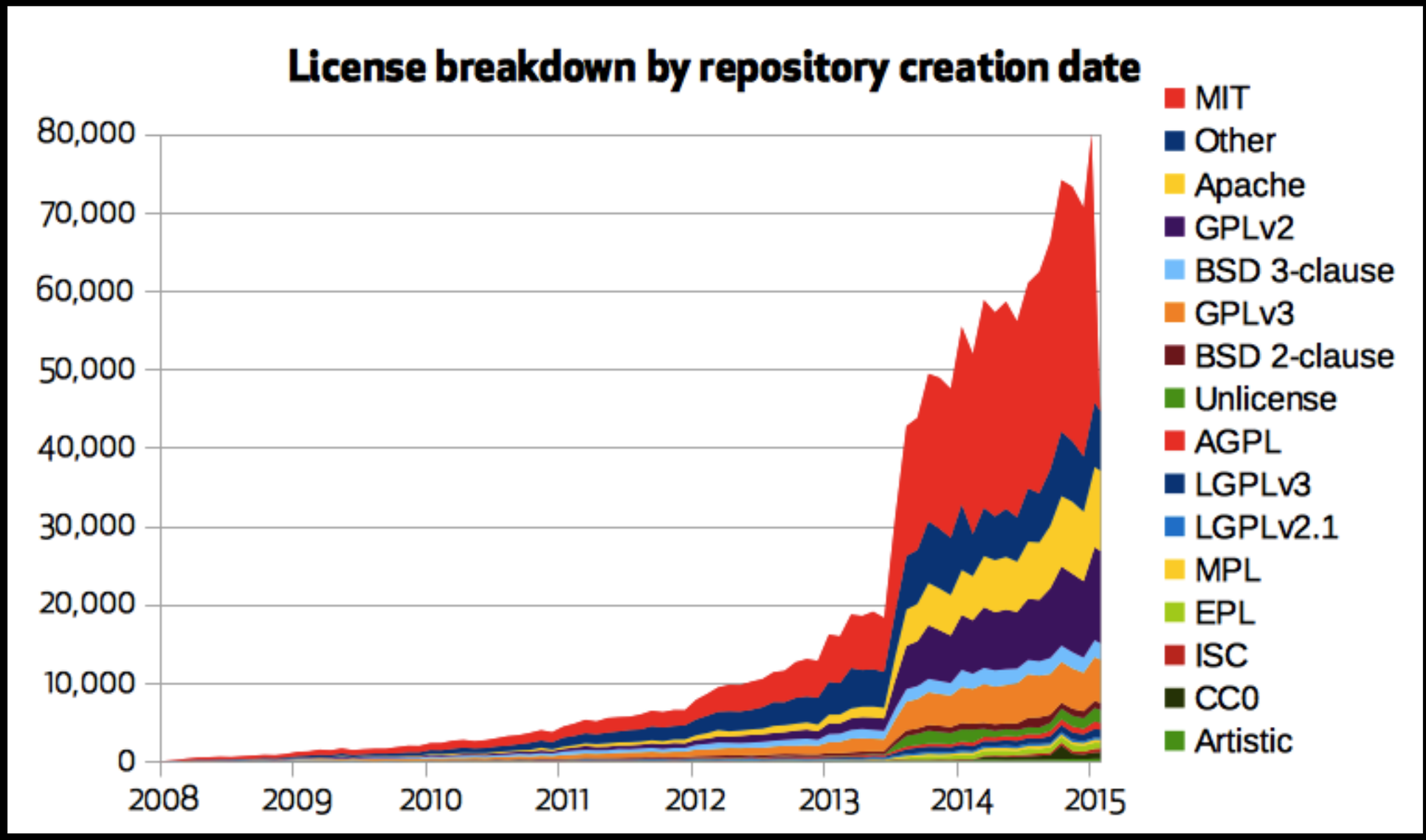
Source: https://github.com/blog/1964-open-source-license-usage-on-github-com

Distributing OSS

Rank	License	% of projects
1	MIT	45 %
2	Other	16 %
3	GPLv2	13 %
4	Apache	11 %
5	GPLv3	9 %
6	BSD 3-clause	5 %
7	Unlicense	2 %
8	BSD 2-clause	2 %
9	LGPLv3	1 %
10	AGPLv3	1 %

Source: https://github.com/blog/1964-open-source-license-usage-on-github-com

Distributing software



Source: https://github.com/blog/1964-open-source-license-usage-on-github-com

License violations

- Automated tools can be used for detecting licensing issues
- Review of source code (including licenses) would typically be part of "due diligence" in the sale of a company
- With violations of open source (copyleft) licenses, you could be taken to court and forced to release the source code
- Topic of research, e.g. *We et al (2017)* on inconsistencies of licensing within OSS projects

Case: React

- Initially developed and used internally at Facebook
- Open Sourced in 2013 under a permissive BSD license, but with a patent clause (BSD+patents)
- Source code is released under permissive BSD license
- Licensees are granted right to any Facebook patents in the software

Case: React

The catch:

“The license granted hereunder will terminate, automatically and without notice, if you […] initiate directly or indirectly, or take a direct financial interest in, any Patent Assertion: (i) against Facebook or any of its subsidiaries[…]"

Case: React

- Increasing concern about the implications of the patent clause
    - ASF put the BSD+patents license on their Category X list
    - Automattic (wordpress developer) decided to drop React
- Perception was that companies relying on React would be at Facebook's "mercy"

Case: React

- React 16 released with MIT license and no patent clause
- Facebook still argues the BSD+patent license is a good solution that can reduce the amount of software patent lawsuits in general

Case: DHIS2 app

- DHIS2 app developed by/for WHO
- Development started before decision was made on license to use for the app
- DHIS2 uses Highcarts - DHIS2 dev team has license -but does not extend to 3rd party developers on the platform
- Highcharts was still used for practical reasons, without considering in detail the license implications

Case: DHIS2 app

Can I use the Non-Commercial License for a government website?

> Governmental or intergovernmental organizations are not covered by the Non-Commercial License. Contact us for assistance in identifying your license needs.

May I use your software under the Non-Commercial License for Open Source Projects?

> Although Highsoft's software have open source codes, our software is not licensed as an open source software and is unfortunately not compatible with any open source software license like Apache 2 or any GPL. See also Non-Commercial Redistribution above.

- App had to be released under GPL 3 - strong copyleft
- **Result**: App had to be re-written (replacing highchairs with chart.js) before it could be released

FOSS and platforms

- Is FOSS relevant in platforms ecosystems?

- Platform openness can be used to encourage development of complements and grow the platform ecosystem

- Balance between opening up to encourage growth, and being too open and not having enough governance mechanisms - risk of exploitation

Source: Karhu et al. (2018)

Boundary resources

- Resources enabling third party development through tools and regulations
- Platform owners must develop boundary resources that:
  - enable innovation, design and development of new functionality to the platform
  - control the platform and its evolution in some desired direction

Platform Forking

- Platform forking: creating a new competing platform from an existing platforms resource base.

- Forking not only the core code base, but also "forking" compatibility with the complements - strategic rather than just technical

- Enabled by sharing of the platform core IPR (i.e. open sourcing)

Source: Karhu et al. (2018)

Case: Android forks

- Karhu et al (2018) study how the Android platform has been forked, and how Google as the platform owner has tried to defend itself against forking

- Define two types of openness of platforms - access and resource

Openness	Boundary resources	Actor who shares	Shared resources	Type of sharing	Platform owner's rational
Access	API, app store	Complementors (e.g. app developers)	Complement, e.g. apps	Shared for distribution	Generate network effects; extract value from complementaries
Resource	Open-source license	Platform owner	Platform core, e.g. AOSP	Shared IPR	Strategic forfeiture of IPR while recovering costs elsewhere

Timeline

Boundary resource

App store

Open-source license

Open-source license App store

API

App store SDK license

Forks

Android Market hacked by CyanogenMOD community

Xiaomi MIUI platform launched

Fire OS platform launched

Fire OS adds cloned Maps API

"Amazon App for Android Phones" launched

Host

Cease-and-decisist (legal action)

Key Android apps become proprietary

Android market becomes Google Play

Google Play Services - APIs extracted from core OS

Developer distribution agreement changed

Boundary resource

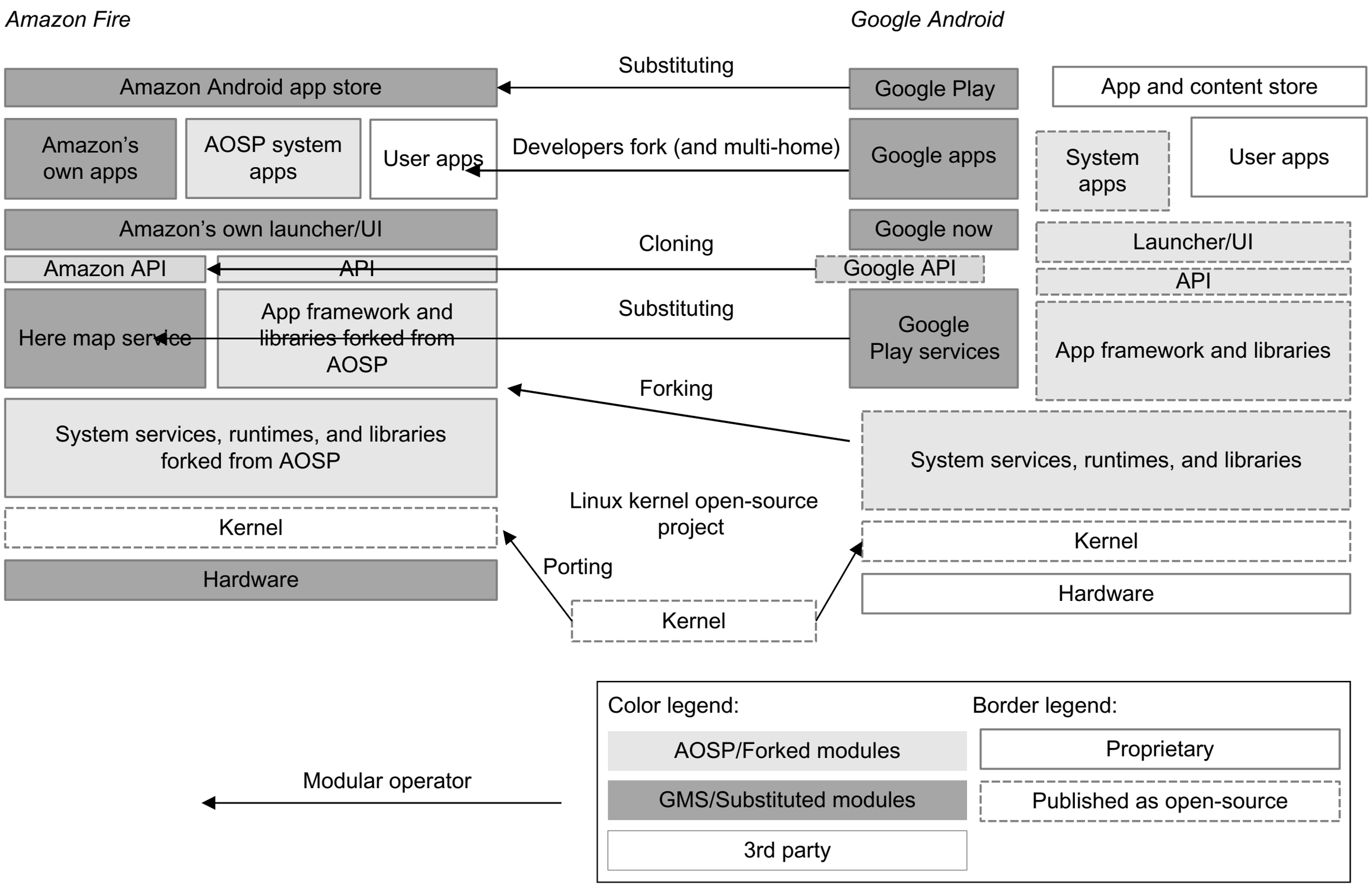
Open-source license

App store

API Client library

Distribution agreement

Figure 2. How Amazon Built Its Fire Platform by Forking Android



Source: Karhu et al 2018

Some key points from Karhu et al

- Platform forking can be seen as a competitive platform strategy that diminishes the host's competitive advantage

- Platform forking is a result of too loose platform governance (example: allowing apps to be distributed to any app store)

- Boundary resources are key to extracting value (rents)

- Boundary resources can be leveraged to "combat the competitive actions of forkers", e.g. withdrawing from openness, contractual arrangements, software designs

Sources

- Feller and Fitzgerald, 2000. *A framework analysis of the open source software development paradigm.*
- Fitzgerald, 2006. *The transformation of open source software.*
- Stallman, 2009. *Viewpoint: Why “open source” misses the Point of Free software.*
- Leister and Christophersen (eds), 2015. *Open Source, Open Collaboration and Innovation.* Mainly chapter 4.
- Karhu, et al, 2018. Exploiting and Defending Open Digital Platforms with Boundary Resources: Android’s Five Platform Forks.
- https://plato.stanford.edu/entries/intellectual-property/
- https://www.patentstyret.no/tjenester/patent/hva-kan-du-soke-patent-pa/
- http://publications.nr.no/1439981439/Compendium_INF5780H15.pdf
- https://github.com/blog/1964-open-source-license-usage-on-github-com
- https://medium.freecodecamp.org/facebook-just-changed-the-license-on-react-heres-a-2-minute-explanation-why-5878478913b2
- https://openedreader.org
- Wikipedia (FOSS)
- https://en.wikipedia.org/wiki/Intellectual_property
- https://en.wikipedia.org/wiki/Permissive_software_licence
- https://en.wikipedia.org/wiki/Affero_General_Public_License