UET 111 · CHAPTER 1 · FOUNDATIONS

Free and Open-Source Software

General Introduction

Dr. Zehir Hatem
Urban Planning Department · S1, 2026/2027



Outline

1. Definitions and History of Free and Open Source Software
2. Free Software vs. Open Source vs. Proprietary
3. Philosophy and Ethical Issues
4. Types of Licenses
5. Conclusion

Learning Objectives

By the end of this lecture, you will be able to:

- **Explain** the difference between free, open-source and proprietary software;
- **Read** a software license and tell what it allows and forbids;
- **Choose** a suitable license for your own maps, data and documents;
- **Recognize** the open tools planners use (QGIS, OpenStreetMap…).

No programming required

Every idea is explained in plain language. New terms are collected in the **glossary** at the end.

Software in a Planner's Day

You use software to...

- draw and analyse maps (GIS);
- edit plans and 3D models;
- write reports, build spreadsheets;
- publish maps on the web.

Each tool has hidden rules:

- May I copy it to a colleague?
- May I adapt it to local needs?
- What happens if the price rises, or the company stops the product?
- Who owns my files?

Key idea

The license decides who controls the tool: you, or its vendor.

What is Software?

Definition

Software = the instructions that tell a computer what to do: your GIS, your web browser, your spreadsheet.

Every program exists in two forms, like a meal:

Source code	Instructions written by programmers, readable by humans. Like a **recipe**
Binary (executable)	The ready-to-run version the computer uses. Like the **cooked dish**

With only the dish, you cannot see the ingredients or change the recipe. Software comes with a license: the rulebook saying what you may do with it.

Free Software: Definition (GNU/FSF)

"Free" as in *free speech*, not *free of charge*

Free Software is software that guarantees the user's essential freedoms, as defined by Richard Stallman and the Free Software Foundation (FSF).

A program is free software if its users have the Four Essential Freedoms:

1. **Freedom 0**: Run the program as you wish, for any purpose
2. **Freedom 1**: Study how the program works, and adapt it to your needs (*access to the source code is a precondition*);
3. **Freedom 2**: Redistribute copies, to help others
4. **Freedom 3**: Improve the program and release your improvements to the public, so the whole community benefits

Source: gnu.org/philosophy/free-sw.html

Open Source Software: Definition (OSI)

Definition (Open Source Initiative – OSI)

Open Source Software is software distributed under a license complying with the *Open Source Definition*: the source code must be available, and the license must allow free use, modification, and redistribution, including for derived works.

Key criteria of the Open Source Definition (10 criteria in total):

- Free redistribution (no restriction on selling or giving away)
- Inclusion of source code
- Permission to create derived works and distribute them
- Integrity of the author's source code

Nearly the same software but different emphasis: **ethics** (FSF) vs. **practical methodology** (OSI).

Source: https://opensource.org/osd

Free Does Not Mean Free of Charge

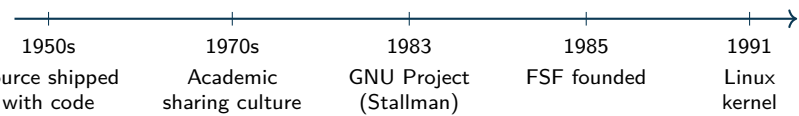
Category	Price	Source code	Modify and share?	Example
Free / open-source	Often gratis, may be sold	Available	Yes	QGIS
Freeware	Gratis	Hidden	No	Google Earth Pro
Shareware	Free trial, then pay	Hidden	No	WinRAR
Proprietary (paid)	Licence fee	Hidden	No	ArcGIS

Careful

Freeware costs nothing but gives you **no freedom**: you cannot study, change or redistribute it.

History: From the Origins to the 1980s

- **1950s–1970s:** Software was delivered **with source code** by manufacturers (IBM, DEC). Sharing was the norm among academic and research communities (early Internet culture).
- **Late 1970s:** Software becomes a **commercial product**. Companies begin closing source code and imposing licenses.
- **1983:** Richard Stallman announces the **GNU Project** ("GNU's Not Unix") to build a complete free operating system.
- **1985:** Creation of the **Free Software Foundation (FSF)** and publication of the *GNU Manifesto*.



History: The 1990s and 2000s

- **1991:** Linus Torvalds releases the **Linux kernel** (the core of an operating system) (re-licensed under the GPL in 1992); combined with GNU tools, it forms GNU/Linux.
- **1998:** After Netscape opens its browser code, the term **"Open Source"** is coined (Christine Peterson; promoted by Eric Raymond, Bruce Perens) to rebrand free software in a more business-friendly way; the **Open Source Initiative (OSI)** is founded.
- **Late 1990s:** The **dot-com era**: Apache web server, Mozilla/Firefox, MySQL, PostgreSQL power the early Web.
- **2005–2008:** Google's **Android** (Linux-based) is open-sourced (2008); **Git** (2005) and **GitHub** (2008) let teams worldwide build software together.
- **2001–2006 (geospatial):** PostGIS (2001), QGIS (2002), OpenStreetMap (2004), OSGeo foundation (2006).
- **2010s–today:** Open source runs most of the Internet, every top supercomputer, Android phones and the mapping tools used by cities worldwide.

Comparison: Key Dimensions

Criterion	Free Software	Open Source (OSI)	Proprietary
Source code access	Yes	Yes	No (closed)
Modify and redistribute	Yes	Yes	Forbidden
Philosophy	User freedom, ethics	Practical quality, methodology	Commercial control
Examples	GNU, Linux, QGIS	Firefox, VLC, PostGIS	Windows, ArcGIS, AutoCAD
Typical cost	Usually free of charge	Usually free of charge	Paid licenses
Vendor lock-in	Low	Low	High

Key takeaway

Free Software and Open Source describe almost the same set of licenses; they differ mainly in their **philosophical justification**. Most well-known programs (QGIS, Linux, Firefox) are **both**.

Proprietary Software: Characteristics

- The vendor keeps **exclusive rights**: you buy a **permission to use**, not the software itself.
- The source code is **hidden**: you cannot study or modify it.
- Typical restrictions, written in the **EULA** (the "I agree" contract):
  - no copying, sharing, or "reverse engineering" (taking the program apart to see how it works);
  - activation keys, copy protection, compulsory online accounts;
  - prices per user, per computer, or by subscription.
- **Advantages (claimed):** professional support, clear responsibility, a coherent suite of tools, a planned roadmap.
- **Drawbacks (criticized): vendor lock-in** (switching becomes costly), no way to check quality or security, unclear control over your data, forced upgrades when support ends.

The Philosophy of Free Software

Core belief (Stallman, paraphrased)

Software should respect users' freedom and community solidarity. Using non-free software puts the developer in a position of power over the user, an unethical social system.

Fundamental ideas:

- **Freedom as a right**: digital tools shape society, control over them is a condition of autonomy
- **Knowledge as a commons**: code is a cultural good, sharing it enriches everyone
- **Cooperation over competition**: thousands of developers contributing to a shared program (Raymond's "scratching an itch" and "Linus's Law": *"Given enough eyeballs, all bugs are shallow."*);
- **Education**: open source is the classroom of computer science

Ethical Issues and Stakes

Arguments in favor of FOSS ethics

- **Transparency:** auditable code ⇒ trust, security verification;
- **Sovereignty:** states and public administrations can audit and control their infrastructure (e.g., Schleswig-Holstein's 2024 move to open source; Munich's LiMux, later reversed);
- **Accessibility:** bridges the digital divide; equal access for developing countries, schools, NGOs;
- **Privacy:** no hidden tracking of what you do when the code can be inspected.

Criticisms and open debates

- **Free-riding:** companies monetizing community work without contributing back (selling a "free" core plus paid extras);
- **Changing the rules:** some firms moved from open to restrictive licenses once big cloud providers profited from their code — where is the ethical line?
- **Maintainer burnout:** unpaid critical infrastructure (a flaw in the widely used OpenSSL tool, 2014; a hidden backdoor found in xz, 2024);
- **AI and open source:** models trained on open code — new frontier of licensing ethics.

Societal Impact

- **Public sector:** open-source policies in many governments (e.g., French *law for a digital republic*, EU *Interoperable Europe Act*), open data and open standards
- **Science and research:** reproducibility requires open code, journals increasingly require it
- **Economy:** a 2024 Harvard study values open source at $8.8 trillion (cost to replace it), every modern tech giant (Google, Amazon, Microsoft) relies on and contributes to open source
- **Education:** students learn from world-class real-world codebases
- **Security:** the 2024 xz-utils incident sparked global reflection on funding and securing critical open-source maintainers

Why Should a Planner Care?

- **Cost and access:** no per-seat fees, so every student, municipality or NGO can use the same tools
- **Durability:** open file formats can still be read in 20 years, whatever happens to a vendor
- **Adaptation:** tools can be translated and adjusted to local regulations
- **Sharing:** you can give your whole toolset, data and method to a neighbouring commune
- **Transparency:** public decisions based on analyses that anyone can re-run

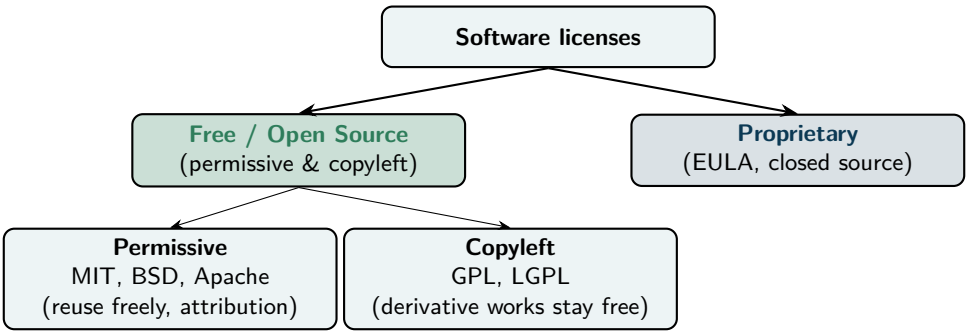
FOSS in Urbanism: The Toolbox

Need	FOSS tool (license)	Proprietary
Desktop GIS	QGIS, GRASS GIS (GPL-2.0+)	ArcGIS Pro
Spatial database	PostGIS (GPL-2.0+)	Oracle Spatial
Formats, projections	GDAL, PROJ (MIT)	FME
Web mapping	Leaflet (BSD-2), MapLibre GL JS (BSD-3)	Google Maps API, Mapbox GL JS v2+
Base map data	OpenStreetMap (ODbL)	Commercial basemaps
Street-network analysis	OSMnx (MIT), GeoPandas (BSD-3)	ArcGIS Network Analyst
Traffic simulation	SUMO (EPL-2.0)	PTV Vissim
3D / CAD	FreeCAD (LGPL-2.0+), Blender (GPL)	AutoCAD, SketchUp

Licenses change over time

Always read the project's `LICENSE`: Mapbox GL JS went proprietary at v2, and the community forked it as **MapLibre** (BSD-3).

Software Licenses: The Big Picture



Three Licensing Attitudes

Family	Attitude	Everyday analogy
Proprietary	"Use it, don't touch it."	Renting a flat: no alterations
Permissive	"Take it, just credit me."	A gift with a thank-you note
Copyleft	"Take it, but keep sharing."	A recipe shared at a potluck: share your improved version too

Copyleft vs. Permissive Licenses

Copyleft licenses

- Derived works must be distributed under the same license
- Guarantees that the software and its derivatives **remain free**
- "Share-alike" principle
- **Examples:** GPL, AGPL (strong), LGPL, MPL (weak)

Permissive licenses

- Allow reuse, modification, and redistribution with few conditions
- Commercial closed-source derivatives are **allowed**
- Main obligation: keep the copyright notice / attribution
- **Examples:** MIT, BSD, Apache-2.0.

$\Rightarrow$ Copyleft protects the **community's rights**, permissive licenses maximize **adoption** (including by companies).

GPL (GNU General Public License)

- Written by the FSF (versions 1989, 1991, 2007): the best-known **strong copyleft** license.

The deal, in plain words:

- You may use, copy, modify and even sell the program
- If you **pass it on** (modified or not), you must give the receivers the **source code** under the **same GPL terms**
- Keep the copyright notice and the license text

Example

A planning office improves QGIS and shares it with partner offices: the partners must also receive the source code and the same freedoms.

Duties start when you **distribute**; private use is unrestricted, The AGPL extends the rule to programs offered through a website.

Used by: Linux kernel, GNU tools, QGIS, PostGIS, WordPress.

LGPL (GNU Lesser General Public License)

- Originally "Library GPL" (1991), renamed "Lesser" in 1999; current version 2007.
- For **libraries** (ready-made building blocks that other programs reuse): a compromise between GPL and permissive licenses.

Key rules:

- A proprietary program **may use** an LGPL library and stay proprietary
- but **changes to the library itself** must be shared under the LGPL
- users must be able to swap in their own version of the library

Used by: GTK, Qt (also GPL/commercial), GEOS (behind PostGIS, Shapely), FFmpeg.

MIT License

- Named after MIT, where it originated in the 1980s; todayś text is identical to the **Expat License**.
- The **simplest and most permissive** widely-used license.

Conditions (essentially one):

- Keep the copyright notice and the license text in all copies
- No requirement to release source code of derivatives
- Commercial use, modification, redistribution: **all allowed**;
- Software provided "*as is*", no warranty

Used by: GDAL, PROJ, OSMnx, React, Rails, jQuery.

BSD Licenses

- Family of permissive licenses from the **Berkeley Software Distribution** (University of California, Berkeley).
- Variants: **3-clause** (common today) and **2-clause** (FreeBSD, drops the non-endorsement clause); original 4-clause now obsolete.

Conditions:

- Retain copyright notice and disclaimer (**attribution**)
- **3-clause:** do not use the author's name to endorse derivatives without permission
- Functionally very close to MIT, the 3-clause variant adds a non-endorsement rule

Used by: FreeBSD, OpenBSD, GeoPandas, Shapely, Leaflet, OpenLayers.

Apache License 2.0

- Written by the **Apache Software Foundation** (v2: 2004), improving on the original BSD-style Apache 1.1.
- **Permissive**, like MIT/BSD, but with significant extra clauses.

Conditions:

- Attribution: keep copyright, license text, and NOTICE file (a text crediting contributors)
- **Explicit patent permission**: contributors cannot later sue you for using their code
- Must **state modifications** made to the code
- Allows inclusion in proprietary products;derivatives need not be open

Used by: Android (AOSP), Kubernetes, TensorFlow, Apache Sedona (geospatial big data).

Summary: License Comparison Table

License	Type	Copyleft	Must share source?	Patent grant	Commercial use
GPL v3	Copyleft	Strong	Required (derivatives)	Yes	Yes
LGPL v3	Copyleft	Weak	Only library changes	Yes	Yes
MIT	Permissive	No	Not required	No	Yes
BSD 2/3	Permissive	No	Not required	No	Yes
Apache 2.0	Permissive	No	Not required	Yes	Yes
Proprietary	Closed	—	Forbidden	—	N/A

Rule of thumb for developers

- Want **maximum community protection** → GPL
- Want **libraries usable by everyone** → LGPL
- Want **maximum adoption** (companies included) → MIT/BSD/Apache
- **Apache 2.0** when you need an explicit patent grant

Creative Commons (CC): Not a Software License

- **Creative Commons** licenses (first released in 2002 by the Creative Commons non-profit) are designed for creative works: texts, images, music, videos, educational material, data.
- They are **not recommended for software** (do not address source code specifics such as linking, patents, or binaries).

The four building blocks:

BY (Attribution)	Credit the creator — always required
SA (Share Alike)	Derivatives under the same license
NC (Non-Commercial)	No commercial use
ND (No Derivatives)	Copy only, no changes

Common licenses: CC-BY, CC-BY-SA, CC-BY-NC, CC-BY-NC-SA, CC-BY-ND; plus **CC0**, a public-domain dedication. NC/ND licenses are not "free" licenses.

Used by: Wikipedia (CC-BY-SA), open educational resources. OpenStreetMap data uses the ODbL (Open Data Commons; share-alike for databases).

Choosing the Right License: Quick Guide

For software:

- End-user application, want community → **GPL**
- Library, want wide use → **LGPL** or **MIT**
- Any code, want zero friction → **MIT/BSD**
- Large project wanting protection against patent lawsuits → **Apache-2.0**

For other content:

- Course slides, photos → **CC-BY** or **CC-BY-SA**
- Total release of rights → **CC0**;
- Research data → specific data licenses (ODbL, ODC-BY)

Practical tools

- **choosealicense.com** (GitHub): guided license chooser
- **Standard short names** (SPDX) like MIT or GPL-3.0-only, so everyone names licenses the same way
- Always include a `LICENSE` file with your project (a *repository* is the project's online folder)

Summary of Key Points

- **Free software**: the four freedoms (run, study, redistribute, improve)
- **Open source** = the same software, justified by practical benefits
- FOSS vs. proprietary: access to source, right to modify, control, transparency, and lock-in are the dividing lines
- A rich **philosophical and ethical** dimension: autonomy, sharing, transparency, sovereignty, with real debates (funding, cloud economics, maintainer sustainability, AI)
- **License families:** copyleft (GPL, LGPL) vs. permissive (MIT, BSD, Apache). Choose according to your goals
- **Creative Commons** for non-software content
- FOSS is the **infrastructure of the modern digital world**

"Free software is a matter of liberty, not price." — Richard Stallman

Glossary

Source code: human-readable instructions of a program.

Binary: the ready-to-run version.

License: the rulebook for using software or data.

Copyleft: "share-alike": modified versions stay free.

Permissive: few conditions, mostly "credit the author".

Derivative work: a modified or adapted version.

Distribution: giving copies to other people.

Library: reusable building block for other programs.

Patent: legal monopoly on an invention.

EULA: the "I agree" contract of proprietary software.

Vendor lock-in: being trapped because switching is costly.

Operating system: the base software (Windows, Linux...).

Check Your Understanding

1. A municipality modifies a QGIS plugin and wants all improvements to stay public. Which license family?
2. A start-up wants to embed your geocoding library in a closed SaaS product. Which licenses allow it?
3. You publish survey data and want attribution only. Which license?
4. Is “freeware” free software? Why (not)?
5. Your commune stores all its maps in a format that only one vendor can open. What risk does this create?