# Open Software Class Discussion

Andrew Nerud

## I. Introduction

Software development is shaped by licensing models that determine access, control, and sustainability. Open source and proprietary software differ in who can modify and distribute code, how developers are paid, and how maintenance is funded. Copylefting adds another layer, ensuring software freedoms remain intact. Understanding these factors is key to evaluating the impact and viability of different software models. This discussion explores these concepts and their role in the software industry.

## II. Defining Open Source Software

Open source software (OSS) allows public access to its source code, fostering transparency, collaboration, and innovation [5]. Companies like Google and IBM actively contribute to OSS, reducing development costs while accelerating progress [8]. OSS adoption has also shifted business models, with companies profiting from services rather than software sales [2]. Some of the most widely used software, such as Linux and Kubernetes, follows this model, enabling businesses to integrate and customize solutions without vendor restrictions [6].

- Real-World Case: Development tools such as git, Apache, and Maven show how open-source foundations enable competition while maintaining shared development efforts [8].
- Discussion Prompt: Should companies that benefit from OSS be required to contribute back to the community? Why or why not?

## III. Defining Closed Source Software

Proprietary software restricts access to its source code, allowing only authorized developers to modify or distribute it [7]. This model ensures security and vendor support, which is critical in regulated industries like healthcare and finance [5]. However, it also limits user flexibility and may lead to vendor lock-in, making companies dependent on specific software providers [2].

- Real-World Case: Microsoft Office remains closed-source, ensuring stability and vendor support, while LibreOffice offers an open-source alternative, highlighting the trade-offs between control and cost [7] [8].
- Discussion Prompt: Should certain industries (e.g., healthcare, finance) be required to use only proprietary software for security reasons? Why or why not?

## IV. What is Copylefting?

Copylefting is a licensing strategy that ensures open-source software and its modifications remain freely available under the same terms [4]. The GNU General Public License (GPL) enforces this, preventing developers from privatizing modified versions and restricting future accessibility [3]. While copyleft promotes long-term openness, some companies avoid it in favor of more permissive licenses like MIT, which allow proprietary use alongside open-source contributions [2].

- Key Takeaways:
  - Pros: Ensures continued freedom, prevents proprietary takeovers.
  - Cons: Some companies avoid copyleft software due to strict licensing [4].
- Discussion Prompt: Does copyleft licensing protect open-source principles, or does it discourage corporate adoption?

## V. Who Owns an OSS's Development Strategy?

OSS is typically developed through a decentralized model, with contributions from independent developers and corporate sponsors [6]. While the original creator may retain ownership, modifications are governed by open licenses, ensuring continuous expansion and iteration [5]. Many large-scale projects thrive due to corporate funding, as businesses rely on OSS for infrastructure while investing in its maintenance [8].

- Real-World Case: Red Hat, a leader in enterprise OSS, was acquired by IBM for $34 billion, raising concerns about whether corporate involvement dilutes open-source values or enhances sustainability [2].
- Discussion Prompt: Should corporations be allowed to control major open-source projects, or does this undermine the community-driven model?

## VI. How Are OSS Developers Paid?

OSS developers earn income through company sponsorships, grants, crowdfunding, and consulting [1]. Many businesses directly fund development to align OSS with their needs while ensuring key projects remain active [5]. Some developers secure full-time employment within tech firms, guaranteeing consistent contributions, while others rely on community-backed funding models like GitHub Sponsors and OpenCollective [9].

- Key Takeaways:
  - Pros: Companies funding OSS ensures sustainability.
  - Cons: Developers working on less visible projects may struggle financially [1].
- Discussion Prompt: Should companies be required to compensate open-source developers whose work they rely on?

## VII. How is OSS Maintainence Funded?

OSS maintenance depends on corporate sponsorships, donations, and revenue models like premium support or cloud-based services [9]. Companies invest in OSS to ensure reliability, benefiting from stable software rather than relying on unsupported versions [2]. Organizations like the Linux Foundation provide structured funding to prevent critical projects from being abandoned [9].

- Real-World Case: The Linux Foundation funds open-source projects like Linux and Kubernetes, with backing from Google, Microsoft, and IBM to ensure stability and security [9].
- Discussion Prompt: Should governments or private companies take responsibility for maintaining essential open-source infrastructure?

## VIII. Final Discussion Reflection

As open-source and proprietary models evolve, the debate over sustainability, corporate influence, and community contribution remains open. Where do you stand?

- Closing Question: Should companies monetize open-source contributions, or does this undermine the purpose of free software?

## References


[1] Christopher Bogart, Christian Kästner, and James D. Herbsleb. How are paid and volunteer open source developers different? a study of the rust project, 2024.

[2] Forbes Technology Council. How open source became the default business model for software, 2018.

[3] Creative Commons. Copyleft — creative commons, open source & free software, 2025.

[4] GNU Project. What is copyleft?, 2025.

[5] Manuel Hoffmann, Frank Nagle, and Yanuo Zhou. The value of open source software. *Harvard Business School Working Paper*, 2024.

[6] F. Petrillo, P. Pires, G. Travassos, and P. Meirelles. Are open source software development communities unaware of software engineering practices? *IEEE Software*, 37(1):12–15, 2020.

[7] EPAM SolutionsHub. Proprietary software: Definition and examples, 2023.

[8] SonarSource. What is open source software? definition guide, benefits & types, 2024.

[9] Tidelift. Open source: who's paying the bills?, 2024.