**Repository Entry**
**Embedded EthiCS @ Harvard Teaching Lab**


## Overview

**Course:** CS 153 Compilers
**Course Level:** Upper-level undergraduate
**Course Description:** Implementation of efficient interpreters and compilers for programming languages. Associated algorithms and pragmatic issues. Emphasizes practical applications including those outside of programming languages proper. Also shows relationships to programming-language theory and design. Participants build a working compiler including lexical analysis, parsing, type checking, code generation, and register allocation. Exposure to run-time issues and optimization.
**Module Topic:** Free Software: Freedoms and Responsibilities
**Module Author:** Trystan S. Goetze
**Semesters Taught:** Fall 2021–22
**Tags:** free software [cs], open-source software [cs], licensing [cs], rights and freedoms [phil], responsibility (duty) [phil]

**Module Overview:** In this module, we consider the question of whether the freedoms protected by free or open-source software come with any ethical responsibilities. After an introduction to the history of free software, students learn about the four freedoms protected by free software licences, such as the GNU General Public License (GPL). We then consider some of the differences between free, open-source, and proprietary software, and why you might choose one licensing scheme over another. We then raise a philosophical question: do freedoms come with responsibilities? In particular, this is framed as an issue of giving back to the community or institution that safeguards those freedoms, so that they can continue to be preserved. Using an analogy with the Tragedy of the Commons, we explore several arguments for and against the idea that users of free or open-source software have an ethical obligation to contribute to those software projects. We also consider whether other institutions, such as big tech corporations or governments, could do the job instead.

Students continue thinking through these issues by discussing a realistic case study that asks them to imagine a dialogue between a software developer and her boss over whether to release a new compiler backend as open-source or to keep it as a trade secret. In the follow-up assignment, students continue to think through these issues by writing a short piece on whether the government should levy a software tax to fund free or open-source projects, or, whether the purpose of a piece of software matters

A typical structure I use in my teaching is to keep the lecture short (20–30 minutes) and devote the remainder of the time to class discussion. This allows students to refresh and add to their knowledge of the material before diving into higher-level learning activities, and leaves plenty of time to structure the discussion with crafted case studies and questions. A shorter lecture is also easier for students to pay attention to all the way through.

| | | |
|---|---|---|
| | when choosing the licensing scheme one publishes it under. | |
| **Connection to Course Material:** | The connection to the course material is made in a few places explicitly, but implicit throughout the lesson is that many popular compilers are entirely or partially free or open-source. In addition, the lecture raises the question of whether anyone should be allowed to own something as fundamental to computing infrastructure as a compiler – but, because compilers are dependent on proprietary hardware on the backend, a conclusion that compilers should be free software may quickly become an argument that *all* software should be free, which is a much more controversial stance. The case study also engages with the course material by imagining a developer who creates a new backend for a compiler to use in creating software for medical devices. One of the assignment questions (on the purpose of the software) may also bring in compilers as one function software may serve that could be relevant in deciding whether to release it as free or open-source. | The topic chosen for this iteration of the module takes inspiration from a subset of the content for the last iteration, developed by Meica Magnani for Fall 2019–20 [repository entry missing]. In that version, there is much more lecture content, which is organized around the question of whether maintainers of free or open-source software ought to be funded. The case study in Magnani's iteration was the HeartBleed bug in OpenSSL, an open-source secure communications software library. HeartBleed was a serious security vulnerability that was not discovered, until it had been exploited by hackers, because the OpenSSL team was overworked and underfunded.<br><br>The decision to focus on responsibilities that may follow from the freedoms of free and open-source software is inspired by Magnani's use of the Tragedy of the Commons in her lecture. The move to this focus is to draw out an ethical issue that directly impacts compiler users, as many compilers are free or open-source. The narrower focus also enables more structured discussion time.<br><br>A suggestion from Arpita Biswas in the Teaching Lab: Another topic that could be explored, which might have an even closer connection to the course material: how the choice of computer language affects the environmental impact of the project. Some languages require additional processing power when being compiled, especially when used for machine learning. For large projects this can make a difference to their electricity usage and carbon footprint. A recent |

| | | |
|---|---|---|
| | | paper on this topic: https://greenlab.di.uminho.pt/wp-content/uploads/2017/09/paperSLE.pdf Summarized in this news article: https://jaxenter.com/energy-efficient-programming-languages-137264.html This topic wasn't adopted because of concerns of overlap with another module that is on environmental impact of computing (specifically, data processing for machine learning models). |

| Goals | | |
|---|---|---|
| **Module Goals:** | 1. Understand the origins and motivations of free and open-source software.<br>2. Know the differences between free, open-source, and proprietary software.<br>3. Understand the freedoms protected by free software.<br>4. Be familiar with and critically engage with arguments for and against the notion that the freedoms protected by free or open-source software come with ethical responsibilities to contribute to those projects. | Finding ethical issues connected to compilers isn't easy! We decided to go via the ethics of free software, since (1) free/open-source software is an important topic that isn't discussed in other modules, and (2) many popular compilers are free or open-source software. |
| **Key Philosophical Questions:** | 1. Do freedoms come with ethical responsibilities (duties, obligations)?<br>2. On whom does the responsibility of maintaining shared community resources fall?<br>3. How do we decide between different value sets, such as the business interests of a company vs. the ethical good the company could do? | These questions follow from a more general philosophical reflection on freedoms and responsibilities in society, inspired by an argument made by Mary Ann Glendon in her book *Rights Talk*. The third question arose organically through discussion with the students, and wasn't explicitly discussed in the lecture or readings. |

| Materials | | |
|---|---|---|
| **Key Philosophical Concepts:** | ● Freedoms / Rights<br>● Responsibilities / Duties<br>● Different domains of values (ethical, business, legal) | Free/open-source software is designed to preserve several freedoms of computer programmers and software users to use the software as they please. These freedoms are akin to rights, and also stand in contrast to traditional approaches to protecting intellectual property rights. |

| | | |
|---|---|---|
| | | Responsibilities, in the sense of duties or obligations, are a fundamental ethical concept. In this context, the question we asked was, are there distinctive responsibilities that arise from particular freedoms? In the case of society, arguably we have responsibilities to contribute to maintaining a society that protects our rights. In the case of free/open-source software, arguably we have responsibilities to contribute to maintaining free/open-source software that we have availed ourselves of.<br><br>The different domains of values arise in the context of actually deciding how best to fulfill the responsibilities one may have regarding free/open-source software. There are legal duties one has if one decides to incorporate some free/open-source code into one's public projects. The business-focused interests of one's employer may conflict with ethical goods involved with contributing more to a free/open-source project. |
| **Assigned Readings:** | ● Richard Stallman, *The GNU Manifesto*, https://www.gnu.org/gnu/manifesto.en.html<br>● Optional reading: Trystan Goetze, "An IP Cheat Sheet" | The Stallman reading is a classical essay in the free software movement, in which Stallman makes his case for free software in general, and the GNU project (an ongoing effort to create an operating system and set of applications and tools released under a free software licence) in particular.<br><br>The supplemental handout was provided to give students a bit of background on the different terms and concepts used when discussing intellectual property, and to reinforce the distinctions between free, open-source, and proprietary software. |

## Implementation

**Class Agenda:**

1. Introduction
2. Origins of Free Software
3. Freedoms of Free Software
4. Distinctions between free, open-source, and proprietary software
5. Do freedoms come with responsibilities?
6. Application to free and open-source software
7. Discussion of the case study
8. Wrap-up and assignment instructions

Timing breakdown:

1. 5 min
2. 5 min
3. 5 min
4. 5 min
5. 5 min
6. 5 min
7. 2 + 20 + 3 + 15 = 40 min
8. 5 min

Total: 75 min

**Sample Class Activity:**

Song Li is a senior developer for Physicker, a healthcare technology company. Her boss, Mavis Sloane, has asked Song to use mainly open-source tools so that the company can save on costs. Song selects an open-source compiler that works well with the languages Physicker's development pipeline prefers.

Song soon finds that some of the hardware used by a few of Physicker's clients isn't very well supported by the compiler she chose, and there aren't good open-source alternatives with a frontend that supports their preferred languages. So, she spends some time developing a new backend for the compiler.

Sloane is impressed, and wants to keep the new backend as a proprietary trade secret. Song, on the other hand, wants to release her work under the same open-source license as the original compiler.

1. What could Song argue when she presents her case to Sloane?

2. What counterarguments might Sloane make in response? How could Song reply?

After a 5-minute introduction from the course head and a 25-minute lecture from the module instructor, the next 40 minutes were spent discussing the following case study. Students took 2 minutes to think about the questions on their own, then 20 minutes in small groups of 3 or 4, then 3 minutes to enter their responses on an online classroom response tool (Padlet), followed by 15 minutes of large group discussion. The module concluded with 5 minutes to wrap up and introduce the follow-up assignment, and for any remaining questions.

**Module Assignment:**

In the style of a blog post (250–300 words), write an answer to one of the following prompts:

Option 1: Should the government levy a tax on proprietary software, and distribute the funds to support free and open-source software? Why or why not?

Option 2: Does the purpose of a piece of software (e.g. medical, accounting, software development) matter when considering whether to release it under a free or open-source license? Why or why not?

"Blog post style" is to set students at ease – i.e., it's not a formal essay.

Providing options for written assignments is always good practice.

Students were told to refer to the Stallman reading, and were encouraged to search for other sources as well.

| | | |
|---|---|---|
| | | A rubric was provided to help students understand how they would be evaluated.<br><br>Students had one week to complete the assignment.<br><br>Grading was done by students of their peers' submissions. Students had 2 weeks to complete the peer review. |
| **Lessons Learned:** | Students listened attentively to the lecture. They were engaged during the discussion, and raised many issues that hadn't been explicitly discussed in the lecture (e.g. power dynamics between employee and boss, business values vs. ethical values).<br><br>I used a website called Padlet to capture their thoughts from the discussion, which was effective in giving us a fall-back resource for continuing the discussion when students were no longer raising their hands. The Padlet was filled with far more notes than we could possibly take up in class.<br><br>No one group was unanimous in thinking that the issues in the case study were easy to solve.<br><br>Pedagogical insights:<br>1. The students who attended were remarkably engaged. This may be an effect of the course being upper-division.<br>2. Attendance was good, but probably less than 50% of registrants. This may be an effect of lecture attendance being explicitly optional for the majority of class sessions.<br>3. Students didn't seem to need/want much time to think individually on the case study before talking in groups.<br>4. Concerns about business values (e.g. profitability, reputation, legal risk) seemed to be well-engrained in the students, and made it sometimes challenging to shift to a focus on ethical values. | Students were interested to learn about the differences between licensing schemes (proprietary, free, open source, open source but closed development, etc.). Additional modules on free/open-source software might be of interest to them. |

## Additional Research Notes

1. While conducting your research for this module, what materials (articles, blog posts, podcasts, etc.) did you find most helpful? Please include one or two sentences by way of explanation if it won't be obvious to a future GF what role the material played in your preparation for the module.

   The most helpful resources I used were Magnani's slides from the previous iteration (available in the Teaching Lab Google Drive) and my own past lecture slides from a unit on intellectual property that I taught in a computer ethics course (some of these are shared in the Google Drive as well). For understanding compilers, in addition to Stephen Chong's opening lecture for CS 153 (which was recorded), I relied on this webpage: https://cs.lmu.edu/~ray/notes/introcompilers/

2. While developing this module, did you have any ideas that were left on the cutting room floor (i.e. ideas about potential topics, readings, activities etc. that were not ultimately incorporated in the module or final repository entry)? If so, please record them here and briefly explain why they did not make the final cut (e.g. time constraints, CS instructor preferences, etc.).

   At the Set A stage, the module had far too much planned content on intellectual property law and foundations, which I condensed into an optional one-page handout instead.

   The initial Set B forms of the case study and assignment were focused on a more general question of whether all compilers or all software should be free. After discussion with the course head, these were refined to be both more realistic and more tightly focused on how the choice between publishing under an open-source or proprietary licence might actually arise in connection with a compiler.

   As mentioned above in the marginalia, Arpita's idea to talk about environmental impacts of different high-level languages is great topic that would be a natural fit for CS 153. Overlap with another module that is substantially similar in theme led to this idea not being developed, but I think it has a lot of promise.

3. A more informal take on lessons learned: What else should a future GF know if we have the opportunity to run this module again? For example, based on the actual performance of the module you may have additional insights or speculations to share – If a class activity was successful, do you think the class size was a key factor? Did you perceive any differences between undergraduate vs. graduate students with respect to receptiveness to the module or success on the assignment? Etc.

   The class size was small (only about half of the 70 registered students attended), which made it easier to circulate in the room when students were in small groups. But I have successfully run similar activities in a class size of 150, so class size isn't a big factor here. Simultaneous enrolment and optional lecture attendance policies may have depressed turnout, despite the course head asking all who could to attend in-person.

4. After reviewing the student feedback form for the module, were there any comments or general takeaways that you think would be useful for future GFs to take into consideration if they are tasked with repeating this module?

   The feedback pool was small (5 responses). Students generally thought the module was delivered well. There was some desire for additional discussion of the differences between alternative licensing schemes – GPL vs. Apache vs. GoLang etc.