Issue 041: Licensing

Adrian Kosmaczewski

February 7th, 2022

Welcome to the forty-first issue of *De Programmatica Ipsum*, dedicated to the subject of *Licensing*. In this edition:

- Graham analyzes the consequences for a society[1] where software is distributed "AS IS".
- Adrian enumerates the historical and hysterical roots[2] of software intellectual property.
- In the Library section[3], Graham reviews the work of Richard M. Stallman[4].

Enjoy this issue! Please subscribe to our free newsletter[5] to stay updated about new releases, share the articles on social media, or contribute[6] if you would like to support our work.

Cover photo by Tingey Injury Law Firm[7] on Unsplash[8].

[1] https://deprogrammaticaipsum.com/fitness-for-purpose/
[2] https://deprogrammaticaipsum.com/the-conquest-of-code/
[3] https://deprogrammaticaipsum.com/category/library/
[4] https://deprogrammaticaipsum.com/richard-matthew-stallman/
[5] https://deprogrammaticaipsum.com/newsletter/
[6] https://deprogrammaticaipsum.com/contribute/
[7] https://unsplash.com/@tingeyinjurylawfirm?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
[8] https://unsplash.com/?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Fitness For Purpose

Graham Lee

February 7th, 2022

Licensing has become the major battleground of the software industry. I do not mean the petty in-fighting between the advocates for copyleft and non-copyleft free software licences, nor the Humpty-dumptyism of the debate over "free software" versus "open source software"[1]. I do not even mean the blood-stained hills where stand the encampments of the open source and the proprietary software warriors. The battle I refer to is the decades-long combat between programmers on the one side, and their sponsors and clients on the other. The war the software industry wages against its own user.

Long ago, before even the Covid-19 pandemic if you can remember back that far, makers of software realised that it would best serve their interests not to sell software, but to take money for it anyway. There were lots of problems with selling things, mostly gathered under the umbrella of "consumer protection". Particularly irksome is the first sale doctrine[2] which says that after you have sold something to a customer, it is theirs to use as they wish. How rude! Do not these busybody lawmakers realise how difficult life is for **checks notes** high salary software engineers?

The software people came up with a wheeze so cunning, it took the media industry associations decades to understand and copy it. The book publishers have been slowest of all to cotton on. Yes we took your money and gave you software, but those two things are unrelated. It is not the same as selling you the software. Having software does not mean you are allowed to use it (as we did not sell it so first sale doctrine does not apply), but in recognition of your donation to our shareholders we will allow you a specific, limited, non-negotiable, non-transferable license to use the software in ways which we dictate. Now we have all the

[1] https://deprogrammaticaipsum.com/issue/issue-021-open-source/

[2] https://www.justice.gov/archives/jm/criminal-resource-manual-1854-copyright-infringement-first-sale-doctrine

upside of taking your money, and none of the downside usually associated with making a sale.

Over time the complexity and selfishness of the end-user licence "agreement" has evolved. Who can forget shrinkwrap EULAs—the licence was written on a sheet of paper inside a sealed box, but by breaking the seal you agreed to the licence? Even though you could not possibly have read it yet! In the internet age, this gave way to the click-wrap licence, where merely launching the installer was sufficient proof that you agreed with the licence terms—not just the terms shown in the installer, but any new ideas the vendor's lawyers might come up with in a fever dream and unilaterally apply.

And believe me, they invented some humdingers. At one point Borland Kylix[3]'s licence granted them the right to enter your property and audit your use of their software[4]. Not just while you were using it either, but for up to a year afterwards. Does it sound like you should be able to challenge this in court? Not so fast—by agreeing to the licence you have also waived your right to a jury trial.

Borland later apologised and made it clear that these conditions should have applied only to corporate customers—which is of course entirely acceptable.

The most risible of these software licence clauses—the one that renders the whole industry a sham, and all attempts to pretend that software engineers are professional, disciplined, or even at all skilled, null and void—is the one true ubiquitous clause. The one idea that everybody in the industry, from the most acquisitive venture capitalist to the most charitable of free software contributors, can agree on.

This is the idea that nobody has the right to expect that any software will do what is claimed, will work properly, or even have the capabilities that were advertised. In more precise legalese:

> The software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose or noninfringement.

That is how the lawyers at MIT phrase it, continuing to say that there is no reasonable expectation of redress through criminal or civil law should the software not do what was claimed, or somehow cause harm in another way. In other words, programmers are so incompetent it is a wonder when software does what you think it ought, and no person or corporation has any reasonable business expecting software to do what they think it should. In *other* other words, even if you did not pay for this software, you have been suckered.

Microsoft are slightly more lenient than open source authors. They do include the exact same clause with almost the exact same wording (in the Windows 11 licence this is section 9, paragraph b), but they do accept that the software should behave "substantially as described in any Microsoft materials that accompany the software"—albeit only for a limited time (one year if you bought Windows, 90 days if you bought a PC with Windows). This may sound like Microsoft are maybe suggesting that their software engineers know what they are doing, but for two issues. Firstly there is *no* substantive description of Windows accompanying the software—you get a USB stick and a shiny hologram. Secondly the licence goes on to limit liability to the purchase cost—if you install Windows 11 and its bugs bankrupt your company, you will get your fifty dollars back and not a red cent more.

[3] https://en.wikipedia.org/wiki/Borland_Kylix

[4] https://forums.anandtech.com/threads/borland-eula-allows-them-to-enter-your-home-and-takes-away-your-right-to-a-jury-trial-update-error-in-the-eula.702576/

This professionalism vacuum is the axle about which revolves a vicious circle of low-quality software engineering. There is no point in improving software quality[5], security[6], or reliability, because there is no harm to the creator in making software that is not fit for purpose. And there is no way that anybody will stake their reputation, in an act of competitive advantage, on the idea that their software *is* fit for purpose because they know how poor software quality practices are.

It is not like nobody is trying to fix this. The entire software engineering machinery, from multi-organisation management initiatives like CMMI[7] to technical practices like TDD[8], is about the management of risk and quality. It is just that these practices are introduced into a culture that optimises for taking neither quality nor risk seriously, and passes responsibility for both onto its customers.

While that "fitness for purpose" clause remains universal, nothing about software can truly change.

Cover photo by Luis Villasmil[9] on Unsplash[10].

[5] https://deprogrammaticaipsum.com/issue-2-quality/

[6] https://deprogrammaticaipsum.com/issue-3-security/

[7] https://www.cmmiinstitute.com/cmmi

[8] https://deprogrammaticaipsum.com/a-brief-and-painful-history-of-programming/

[9] https://unsplash.com/@villxsmil?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

[10] https://unsplash.com/s/photos/broken?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

The Conquest Of Code

Adrian Kosmaczewski

February 7th, 2022

In June 1976, Li-Chen Wang[1] published a new version of Tiny BASIC, making it free to copy for all hobbyists in the Homebrew Computer Club[2], including Bill Gates. The license text of that Tiny BASIC for the Altair was "©Copyleft–All Wrongs Reserved", one of the earliest examples of the real struggle felt by early practicioners trying to understand how to distribute software properly, legally, and maybe even, profitably.

Why did Li-Chen Wang explicitly mention Bill Gates in his "Copyleft" disclaimer? Mr. Gates, who was back then General Manager of a small company called Micro-Soft, had openly called those same Homebrew Computer Club hobbyists "thieves" in a public letter published in page 3 of Volume 1, Issue 9 of the Computer Notes (published in February 1976), a short news pamphlet distributed to Altair enthusiasts. You can read it online[3], thanks to the Internet Archive and its terms of use[4].

"Theft." A statutory offense frowned upon since the dawn of time. Mr. Gates asserted ownership of the code he had written. What does that even mean? If running a program means to literally copy its bit patterns onto the array of memory circuitry in a computer, is not code execution a form a plagiarism?

[1] https://en.wikipedia.org/wiki/Li-Chen_Wang
[2] https://en.wikipedia.org/wiki/Homebrew_Computer_Club
[3] https://archive.org/details/Computer_Notes_1976_01_09/page/n1/mode/2up
[4] https://archive.org/about/terms.php

Sex Pistols

If we believe what he publishes in his Gates Notes[5] every week, Bill is an avid reader these days. But in 1976 he had certainly not yet read anything from Pierre-Joseph Proudhon[6], who in 1840 published a seminal book called "Qu'est-ce que la propriété ? ou Recherche sur le principe du Droit et du Gouvernement", which starts with a damning assertion on its first page:

> Pourquoi donc à cette autre demande, Qu'est-ce que la propriété ? ne puis-je répondre de même, C'est le vol, sans avoir la certitude de n'être pas entendu, bienque cette seconde proposition ne soit que la première transformée ?

For those who do not speak French: Proudhon argues in this book that property is nothing but theft[7]. To be honest, he was not the first to say such a thing: Rousseau and the Marquis de Sade uttered similar thoughts before him; as Saint Ambrose[8] said: *superfluum quod tenes tu furaris* ("the superfluous property which you hold you have stolen").

This idea, that property is theft, was first hauled and later rebutted by Marx. It was later reworked by Proudhon's disciple Mikhail Alexandrovich Bakunin[9], for whom the means of production were to be owned collectively by society as a whole.

Most importantly, Proudhon's and Bakunin's later ideas inspired Pyotr Alexeyevich Kropotkin[10]'s thoughts in his classic book "The Conquest of Bread"[11], where he proposed a decentralized economic system based on mutual aid and voluntary cooperation.

> (...) many critics will claim that people are lazy and they would not work willingly, even if it is only for five hours for the necessities. Kropotkin counters by saying that people are willing to work in jobs they enjoy and given the necessary free time to work on their own, with the guarantee of material stability, people will work willingly on collective gardens or in collective garment factories. (Source: Wikipedia[12])

Collective gardens or garment factories. That is cute. Kropotkin should have mentioned SourceForge and GitHub instead.

Do the ideas of Bakunin and Kropotkin sound familiar? Let us recap: voluntary cooperation; decentralization; means of production owned collectively, and more. Any similarity to your preferred Open Source project is just a coincidence.

Yes, a coincidence; because Anarchy, as a political and philosophical movement, is based on two major principles: circled "A" graffitis on major cities, and the abolition of money. There is a chance your aforementioned favorite Open Source project is making money for somebody; a lot, even. Heck, Red Hat made 34 billion dollars out of Open Source. If that is not a lot of money, I do not know what is.

But if your favorite Open Source project is not making any money, beware, because the author might sabotage the next version[13] of that code. That is what Kropotkin was talking

[5] https://www.gatesnotes.com/
[6] https://en.wikipedia.org/wiki/Pierre-Joseph_Proudhon
[7] https://en.wikipedia.org/wiki/Property_is_theft!
[8] https://en.wikipedia.org/wiki/Ambrose
[9] https://en.wikipedia.org/wiki/Mikhail_Bakunin
[10] https://en.wikipedia.org/wiki/Peter_Kropotkin
[11] https://en.wikipedia.org/wiki/The_Conquest_of_Bread
[12] https://en.wikipedia.org/wiki/The_Conquest_of_Bread
[13] https://www.theregister.com/2022/01/10/npm_fakerjs_colorsjs/

about in terms of "guarantees of material stability" above; an aspect not really being taken care of by the current state of affairs.

But, wait. How did this article go from the Homebrew Computer Club to 19th century anarchists in a few paragraphs? The reason is that the key issue in the discussion of software licensing centers around the concept of *property*. Put in other terms: who owns the software?

If I write an app in Rust[14] and I give you a copy of the final binary, what do you own? What do I own? Can you run it? If so, how much, when and where? What if I give you the source code? What can you do with it? Can you compile it, or are you just supposed to read it? What if the app is written in Python[15], where essentially there is nothing else but the source code? What if I write a SaaS application and host it on AWS? What do you own? What do I owe my customers for their monthly subscription? What does Jeff Bezos own? (Spoiler alert: at the current rate, he will soon own everything there is to own.)

All these are valid and difficult questions. They need an answer, because we do not live in that parallel universe where money does not exist, or where the Sex Pistols do not need to release a song called "Anarchy in the UK"[16]; in this universe it does and they do, and money has a lot of power.

And because of this power, we need to know who is responsible for what. For example, what if a small business owner bought an Apple II in 1980 and installed VisiCalc in it, but later went bankrupt? Would Software Arts or Dan Bricklin[17] have been responsible for such an event? Let us not venture into actual tragedies, where lives where at stake during (or affected because) the execution of a piece of code whose license dismiss all liabilities (more on this issue later).

Hence, copyrights, patents, and licenses.

Origins

The truth is that, in 1976, very few people knew what software was; let alone what a software license should look like.

Page 16 of Byte Magazine's "Bicentennial Issue"[18], published in September 1976, features an article by Calvin Mooers called "Are You an Author?", also freely available[19] on the Internet Archive. The article explains in detail how copyright works, and what legal framework exists to protect intellectual property in the form of software.

Clearly, Bill Gates' letter, published a few months earlier, had triggered many spirits. Mr. Mooers, who had been talking about the subject for a while[20], mentioned the word *"softlifting"* for the crime (yes, crime) of illegally using or distributing software.

Mr. Mooers' article references prior work: one of those is "Development of an international system for legal protection of computer programs"[21] (Communications of the ACM, April 1976, Vol. 19 No. 4, Pages 171-174) by a certain Oliver Smoot. This one explicitly mentions

[14] https://deprogrammaticaipsum.com/the-great-rewriting-in-rust/
[15] https://deprogrammaticaipsum.com/issue-35-python/
[16] https://en.wikipedia.org/wiki/Anarchy_in_the_U.K.
[17] https://en.wikipedia.org/wiki/Dan_Bricklin
[18] https://archive.org/details/byte-magazine-1976-09
[19] https://archive.org/details/byte-magazine-1976-09/page/n17/mode/2up
[20] https://dl.acm.org/doi/10.1145/356643.356647
[21] https://cacm.acm.org/magazines/1976/4/11479-development-of-an-international-system-for-legal-protection-of-computer-programs/abstract

the need for a "fair use doctrine" (page 3), particularly in the case of developing countries, many of which did not even have a single computer inside their borders, but would, and soon enough.

Even the United Nations discussed the issue of software licensing, patents, and copyright, in the framework of equal development opportunities for all countries. "The Application of Computer Technology for Development, Report of the Secretary-General"[22], published in May 1970, says in page 63 ("The Protection of Software"):

> It can be argued that computer programs like mathematical algorithms are ideas and as ideas they are not patentable. (...) no national patent law (with the exception of that of France of 1968, which came into force in 1969) makes specific reference to computer programs.

Precisely, France's 1968 law 68-1 about patents[23] literally does not consider "les programmes d'ordinateurs" as patentable inventions. This has been changed since, but it set a strong precedent worldwide, as anything that France does in general.

However, the overall ignorance of lawyers and judges in technological matters during the past 50 years has been, and is, legendary[24]:

> Whether you agree with what the Supreme Court said or ultimately ruled, it is impossible to ignore the reality that the Court just didn't understand computers, and certainly didn't understand the invention in question. A trend that runs throughout all Supreme Court opinions dealing with software and most Supreme Court decisions that deal more generally with patent matters.

Great. The theft of software, a statutory offense, is to be judged by people blissfully unaware of the nature of the thing being stolen. What could possibly go wrong?

Laws

The dramatic rise of software industry behemoths in the American economy, their active lobbying in Washington, and the overall ignorance of the field by those creating and applying laws, led to a series of successful changes in legislation. Successful, for the software industry, that is.

For those interested in the current state of things, chapter 5 of "Geekonomics: The Real Cost of Insecure Software"[25] (2007) by David Rice is a must read, dedicated to the subject of liability in software projects. Or rather, the lack thereof, as guaranteed by said legislation.

After an explanation of the inner workings of the American legal system, one of the core points of the chapter is the all too important distinction between *negligence* versus *strict liability*:

> (...) parties engaged in risky activities must be willing to ensure the safety of others as a price of doing business. (...) Software manufacturers might indeed enjoy absolute immunity, but as history has shown, it will not, and should not, last forever. (...) Zollers believes the court's reasoning on strict product liability for (airplane) charts is a more perfect analogy for understanding software rather

[22] https://eric.ed.gov/?id=ED046461

[23] https://www.legifrance.gouv.fr/loda/article_lc/LEGIARTI000006281073/1979-07-01/

[24] https://www.ipwatchdog.com/2014/11/30/the-history-of-software-patents-in-the-united-states/id=52256/

[25] https://www.informit.com/store/geekonomics-the-real-cost-of-insecure-software-9780321477897

> than the traditional thinking of copyright and intellectual property. Namely, software, like navigation charts, is functional and not literary. (Pages 211-220)

"The Success Of Open Source"[26] (2005) by Steven Weber is a bit of an outsider book in our industry. Written by a political scientist instead of a technologist, it rightly analyzes the social underpinnings of Free and Open Source software as "an experiment in social organization around a distinctive notion of property rights" (page 227). Mr. Weber starts by explaining the issues of property in software, and concludes:

> The license represents foundational beliefs about the constitutional principles of a community and evolving knowledge about how to make it work. (page 185)

The book finally puts the GPL in its right place in the great book of history, as one of the most important breakthroughs in the history and philosophy of property:

> Open source licenses generally depend on copyright law for their claim to enforceability. (...) The open source process relies on the principle that all software is always a beta release. (...) The binding of third parties to an agreement like this (the viral clause of the GPL) is more like a property right, just one configured around distribution rather than exclusion. (Pages 209 to 213)

Through research, I found out that the complex issues of software licensing are conveniently ignored by many authors, including McConnell[27] and Barry Boehm. These issues are better explained by authors versed in politics and law, and we, as technologists, should read more of those. An important exception to this rule is Dr. David Wheeler, who has simplified our understanding of licenses, creating a map of all of them[28], explaining the connection between FLOSS and commerce[29], and even naming the first copyleft license by Richard Stallman[30] among the most important software inventions[31]:

> The first real copylefting license (the Emacs Public License) was developed by Richard Stallman in 1985 – but since copyleft is really a social and legal invention, not a technological one, it's not included in this list.

Enough Is Enough

Did you know that the Computer Fraud and Abuse Act[32] of 1986, amended in 2001, "explicitly forbids action against software manufacturers for negligent manufacturing of software", and that the provisions of this same act are at the origin of the tragic and all too early loss of Aaron Swartz[33]?

Did you know that the DeWitt Clause[34] contained in most commercial software licenses literally forbids anyone from publishing information about products without the owner's consent? A clause that is a *de facto* infringement in the famous freedom of expression that Americans are so happy to talk about all the time.

[26] https://www.hup.harvard.edu/catalog.php?isbn=9780674018587
[27] https://deprogrammaticaipsum.com/steve-mcconnell/
[28] https://dwheeler.com/essays/floss-license-slide.html
[29] https://dwheeler.com/essays/commercial-floss.html
[30] https://deprogrammaticaipsum.com/richard-matthew-stallman/
[31] https://dwheeler.com/innovation/innovation.html
[32] https://en.wikipedia.org/wiki/Computer_Fraud_and_Abuse_Act
[33] https://en.wikipedia.org/wiki/Aaron_Swartz
[34] https://dwheeler.com/essays/dewitt-clause.html

We must, as a society, end the reign of the EULA, at least in its current form. This will not happen without government intervention, which this magazine openly encourages and support. It is time for software companies to face their responsibilities, and if needed, to face prosecution, in proportion to the preeminence of software in our modern world.

Simply put, current software licenses reserve all wrongs to society and all profits to Big Tech[35]. It is time for a change.

Back To Micro-Soft

Microsoft[36] has always had a complicated relationship with Free and Open Source software. Well, no, scratch that. It is not actually complicated. The key issue was always money. These days Microsoft is happy to use, collaborate in, and release software under open source licences; as long as their bottom line grows, all is fine.

And boy did it grow since Ballmer left the helm.

While he was still there, however, things were much more rocky. First he argued that Linux was communism[37]. One year later, he said that Linux was cancer[38]; well, *actually,* Linux was announced in late August 1991[39], which means that it is technically a Virgo.

In 2002, under pressure from pretty much everyone around them, Micro-Soft begat "Rotor"[40], an incomplete, clunky, cross-platform[41] version of .NET 1.0 released under a "Shared Source License"... whatever that was.

Then, *coup de théâtre:* the cover story of the February 2007 issue of Dr. Dobb's Journal was titled "Microsoft Loves Linux: What's With That?" (sadly not available online). In it, Michael Swaine explained the repercussions of the recent Microsoft-Novell agreement, another chapter in the whole SCO vs Novell saga of the 2000s[42]:

> But instead, the reaction to the announcement was all about something that Microsoft was giving to Novell, or rather to Novell customers, under the agreement: A promise not to sue them for intellectual property infringement. (...) As for the Microsoft-Novell deal, it may have, by contrast, helped Red Hat's image. Red Hat's stock went down after these developments, but it may come out a winner in the long run.

Now here is a prediction that turned out to be true; this explains each and every one of the 34 billion dollars.

That same year 2007, Ballmer said that Free and Open Source Software (FOSS) violated 235 Microsoft patents[43]. But then in 2009, Microsoft "stunned the Linux World" by submitting device drivers[44] to the Linux Kernel... with a GPLv2 license.

[35] https://en.wikipedia.org/wiki/Big_Tech
[36] https://deprogrammaticaipsum.com/issue-37-microsoft/
[37] https://www.theregister.com/2000/07/31/ms_ballmer_linux_is_communism/
[38] https://www.theregister.com/2001/06/02/ballmer_linux_is_a_cancer/
[39] https://groups.google.com/g/comp.os.minix/c/dlNtH7RRrGA/m/SwRavCzVE7gJ
[40] https://en.wikipedia.org/wiki/Shared_Source_Common_Language_Infrastructure
[41] https://deprogrammaticaipsum.com/issue-19-cross-platform/
[42] https://en.wikipedia.org/wiki/SCO_Group%2C_Inc._v._Novell%2C_Inc.
[43] https://www.macobserver.com/tmo/article/Ballmer_FOSS_Apps_Violate_235_Microsoft_Patents
[44] https://archive.nytimes.com/www.nytimes.com/external/idg/2009/07/20/20idg-microsoft-stuns-linux-world-submits-source-code-for-65456.html

In 2016, years after leaving Microsoft, Ballmer finally openly confessed his love for Linux[45] while Microsoft joined the Linux Foundation[46]. As a result, Visual Studio Code (MIT License[47]), .NET (MIT License[48]), TypeScript (Apache License 2.0[49]), F# (MIT License[50]), and even SQL Server[51] (ah, sorry, not open source), they all run in Linux these days. And if that was not enough, Linux is running on Azure[52], and you can run Linux apps natively on Windows[53].

All is well that ends well.

Maybe we should start a BreadTube[54] channel for De Programmatica Ipsum. But before that happens, remember to donate to your indie-run, understaffed, but incredibly useful favorite FOSS project; and even to this magazine[55], if you enjoy it. We must all help each other to make a living out of our work. And instead of a license, a copyright, or a legal notice in your code, consider adding a blessing, like the one in the SQLite source code[56]:

> May you do good and not evil. May you find forgiveness for yourself and forgive others. May you share freely, never taking more than you give.

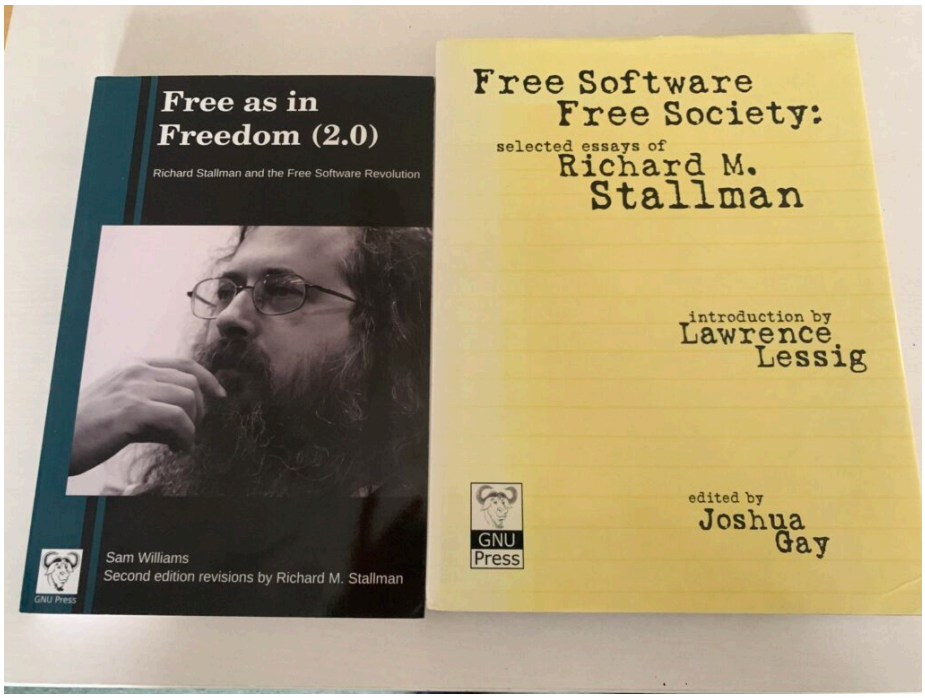
Cover photo by mia swerbs[57] on Unsplash[58].

[45] https://www.zdnet.com/article/ballmer-i-may-have-called-linux-a-cancer-but-now-i-love-it/
[46] https://www.extremetech.com/computing/239616-hell-freezes-microsoft-joins-linux-foundation
[47] https://github.com/microsoft/vscode/blob/main/LICENSE.txt
[48] https://github.com/dotnet/runtime/blob/main/LICENSE.TXT
[49] https://github.com/microsoft/TypeScript/blob/main/LICENSE.txt
[50] https://github.com/fsharp/fsharp/blob/master/LICENSE
[51] https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-overview?view=sql-server-ver15
[52] https://www.wired.com/2015/09/microsoft-using-linux-run-cloud/
[53] https://docs.microsoft.com/en-us/windows/wsl/tutorials/gui-apps
[54] https://en.wikipedia.org/wiki/BreadTube
[55] https://deprogrammaticaipsum.com/contribute/
[56] https://github.com/sqlite/sqlite/blob/master/src/main.c
[57] https://unsplash.com/@miaswerbs10?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText
[58] https://unsplash.com/s/photos/graffiti-anarchy?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText

Richard Matthew Stallman

Graham Lee

February 7th, 2022



It would be inappropriate to have an issue on software licensing without including one of the people whose work has done most to shape the topic. Somehow we managed not to mention him by name in the issue on Free, Libre, and Open Source Software[1]. Well, today we correct that.

RMS is, of course, a prolific writer of software (Emacs[2], GCC[3], Texinfo[4], GDB, GNU Make[5]), of copyleft licenses (the GPL[6], LGPL[7], GFDL[8]), emails, and awkwardly pedantic, sometimes plain offensive missives regarding whether words are being used with the meanings he perceives to be correct. For the purpose of this article I will focus on two of his books: "Free as in Freedom (2.0)"[9], and "Free Software, Free Society" (third edition)[10].

Douglas Hofstadter[11] would recognise "Free as in Freedom (2.0)" as a metacircular reference. It is a book about RMS, copyleft, and intellectual freedom, that only exists because of RMS, copyleft, and intellectual freedom. Very much a strange loop, even in a particularly

[1] https://deprogrammaticaipsum.com/issue-21-open-source/
[2] https://www.gnu.org/software/emacs/
[3] https://gcc.gnu.org/
[4] https://www.gnu.org/software/texinfo/
[5] https://www.gnu.org/software/make/
[6] https://en.wikipedia.org/wiki/GNU_General_Public_License
[7] https://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License
[8] https://en.wikipedia.org/wiki/GNU_Free_Documentation_License
[9] https://static.fsf.org/nosvn/faif-2.0.pdf
[10] https://www.gnu.org/doc/fsfs3-hardcover.pdf
[11] https://deprogrammaticaipsum.com/douglas-hofstadter/

self-referential community that has self-expanding acronyms (GNU's Not Unix) to multiple depths (Hurd stands for Hird of Unix-Replacing Daemons, where Hird stands for Hurd of Interfaces Representing Depth).

"Free as in Freedom" was a biography of Stallman, written by Sam Williams and published in 2002 by O'Reilly and Associates at a time when they were still somewhat supportive of the free software movement. Crucially it was release by Williams under the terms of the GNU Free Documentation License, allowing others to create and share derived works.

An aside here—the GFDL tried to do for prose what the GPL had done very successfully for software, creating an intellectual commons whose shared nature was protected by subverting the same copyright restrictions often used to alienate creative works. GFDL was created in 1999, shortly after the Open Content[12] and Open Publication Licenses[13] and a few years before the first Creative Commons[14] licenses. Today the CC licenses are generally seen as easier and more popular choices for non-software creative works, though some software is released under CC licenses (for example any code quoted in Stack Overflow).

One reader of "Free as in Freedom" who had corrections to distribute was Stallman himself, who rewrote the emotional tone of some sections, corrected technical discussions, and supplied his own perspective on events described remotely in the original. Where "iWoz"[15] is an autobiography in the words of another writer, "Free as in Freedom (2.0)" is a biography incorporating the experiences, analysis, and feelings of its subject.

Many of the stories about Stallman are quintessential RMS brand—arguing with journalists over the term "Linux" where they mean "GNU/Linux", using the first hexadecimal protest chant at a rally outside Lotus' offices; early disussions of the GNU General Public License having to distinguish between free "zero cost" and free "zero restriction" (RMS does not mind software being sold for money, as long as the transaction is freedom-preserving). Some are surprising—like your author, RMS was a keen folk dancer for years, until an injury forced him to stop.

All the tales in this book paint a picture of a driven, influential person who changed the direction of the software industry. To see the direction of that change and its impact, look to the other of this month's works, "Free Software Free Society".

This book is a collection of essays by RMS on the philosophy and development of Free Software, around and through the GNU system of Unix-replacing software components. It is a wide-ranging work, broadly covering the false idea of "intellectual property" which actually describes a number of different, government-imposed limitations on intellectual freedom, sometimes falsely dressed up as "rights". Two of these limitations—copyright and patent monopolies—are covered in depth (the other two limitations that the essays do not deal with are trademarks and trade secrets).

Copyleft is a deliberate subversion of copyright using its own mechanism, and particular attention is paid to the ways in which copyright licenses (and particularly copyleft licenses) can enable and protect intellectual freedom among the software-using community, which as software has "eaten the world" has grown to encompass everyone indirectly and a good chunk of the world's population directly. By the end of part 7 ("Value Community and Your Freedom") you see the political viewpoint of the author: one that seeks to remove power

[12] https://www.opencontent.org/
[13] https://opencontent.org/openpub/
[14] https://en.wikipedia.org/wiki/Creative_Commons
[15] https://en.wikipedia.org/wiki/IWoz

relationships and injustices that are mediated by computers and software; one that values informed democracy and sees the liberation of information as a route to achieve it.

Of course, Stallman being Stallman he chooses his words carefully and makes sure you know why he has chosen them. So you will also find out why Open Source "misses the point", why Apple's products do not deserve to be called by the names Apple bestows upon them, and when it is or is not appropriate to use words like "marketplace" and "ecosystem" in a software context.

But mostly, you will find the development of the philosophical paradigm behind a movement that grew from a text editor that you could buy on tape for $150, to a global reshaping of software businesses and the ways in which software is produced, used, and paid for. The campaign is not over—software freedom has struggled to react to SaaS, where no software is distributed so no copyright licenses get triggered—though many victories have occurred along the way.

Cover photo by the author.