Spring 2022

Software as a Tool not a Master

Dalton Leif Lange

Software as a Tool not a Master

An Examination of User's Liberties in Software Design

Dalton Leif Lange

June 12, 2022

Consider for a moment the hours of your life wasted on YouTube, Twitter, or Facebook. Or that time you almost clicked the purchase button when playing that app on your phone. That wasn't a mistake, the developers of this Software have designed it so you stay on their app, or accidentally push the purchase button. Software should be a tool that we, the user of the software, control, but instead the opposite is often true. When the user doesn't control the software, then the software controls the user when they use it, and since the developer controls the software, ultimately the developer controls the user [1].

Most will agree that our ultimate goal should be to produce the best software for the largest number of people. The "best software" should be software which serves the user, and not software which the user serves. In order to determine what progress we've already made towards this goal, and what our next steps should be, it is worth examining the short history of software development through the lens of user's liberties.

1 The Era of Shareable Software

The modern craze for the computer started in the 1940s, with developments at University of Pennsylvania on the Electronic Numerical Integrator and Calculator (ENIAC) [2]. This incredible machine was programmed by plugging and unplugging cables, as well as adjusting various switches, and is one of the first examples of the creation of "software." The user of the computer was in complete control of the software on it, it was as much a tool as a calculator is.

Computers were complex, but the scope of problems they could solve was small. They were used to solve laborious calculations, running census data, calculating the trajectory of missiles, and similar. As such, the "software" running on these machines was simple, and sought to serve the academics and businesses that were programming them.

Because software was simple in these times, it was not considered very valuable. This encouraged those in the business of creating software to share their programs, modify them, and share them more. Richard Stallman, an academic in those days, commonly gives the analogy of a cooking recipe to communicate how software was understood during this time. Step-by-step instructions on how to complete a specific task, that a skilled chef could change to their liking, "Add some mushrooms, [because] you like mushrooms. Put in less salt because your doctor said you should cut down on salt–whatever."[3]

It is easy to see how the user is in complete control in these days. If the mushrooms are bothering you, remove them. If it makes more sense to add sugar, add the sugar. The user was in complete control of the tool they were using, and if your tool helped you, it was bound to help someone else, thus you shared it around.

But shareable software's days were numbered. By the 1970s software began getting more complicated. As software got more complicated, its value increased, and it is easy to imagine that if software got too valuable, people would stop wanting to share it. Richard Stallman recounts an anecdote, which he says is what made him realize the shareable software culture was changing.

It was a normal day in the early 1970s at Massachusetts Institute of Technology's Artificial Intelligence Laboratory (AI Lab) where Richard Stallman worked. Recently, the laboratory had been gifted a new printer by the company Xerox [3]. This printer had a problem though, it was prone to frequent paper jams. Unable to fix the hardware issue plaguing the machine, Stallman reached for what he knew: software. It would not be hard to have the machine check if it had jammed, and send a signal out to everyone who had a print job to go check it. A marvelous solution for the time.

All Stallman had to do was identify who had the code for this machine. Software is written as step-by-step instructions in a human readable language, which we then transform into computer readable language. Stallman had the computer readable program for the printer, but not the human readable one, which he could modify. We often call the human readable one "source code," since it is the "code" which acts as the "source" for the program. Luckily for Stallman it wouldn't be much of a problem to obtain the source code. Someone must have the source code and–in the spirit of the time–be willing to share a copy.

A few months later, Stallman would catch word of a professor at Carnegie Mellon University, who had helped Xerox program the printer, and was still working on it for research, thus he must have a copy of the source code. Stallman found a good enough reason to visit Carnegie Mellon, and asked the professor if he could have a copy of the source code for the printer. To Stallman's surprise, the professor simply said "no." They elaborated that they were under a Nondisclosure Agreement (NDA) and had agreed not to give anyone a copy of the source code.

An NDA is a contract which legally restricts someone from communicating about a given topic or idea. The point of it is to ensure a company's secrets aren't revealed to competitors, helping to ensure their advantage. Thus is goes against the very nature of the shareable culture for software that had been cultivated for the past three decades.

This event shook Stallman to the core. While it does not seem out of place in our modern day, back in the 70s it was a strange interaction. Akin to asking your neighbor for sugar only for them to proclaim they could not give you their sugar because their boss said so.

The next decade (from the 70s to the 80s) would see a giant shift in culture, as NDAs became more common, and individuals became more willing to sign them. This was when computers would begin to be marketed to people outside of business, and research. Everyone could benefit from a computer, and suddenly the software on that computer was incredibly valuable, far too valuable to be given away freely.

But Stallman remained vigilante through the decade. Many companies attempted to hire him, but they all wanted him to sign an NDA, so he refused. He could not bear the thought of having to refuse to give a copy of source code to someone else. This left Stallman ostracized and alone as the old era of shareable software left, and the new era of what would come to be known as "proprietary software" was ushered in.

2 The Era of Proprietary Software

It was the 80s, and between Apple, IBM, Commodore, and Radio Shack civilians were being offered incredibly powerful computers they could use in their own home [2]. Each computer had its own operating system, with its own programs, and its own strengths and weaknesses. And most importantly for us, all the programs on these computers were "proprietary". You could not get the code for MS-DOS, AmigaOS, or Classic Mac OS. If you had a problem, you could not create your own solution, you needed to ask the company which produced your computer to do it for you.

This wasn't a terrible infringement on users liberties. Most of the software being produced had the specific goal of being a good tool for the user, but since the source code was too valuable we lost something. Users could no longer modify the program to suit their needs. This meant there were lots of people with problems they wanted fixed, and people who had the skills to fix these problems, and yet everyone had to suffer through it together, helpless.

Richard Stallman identified this problem while reflecting on the previous decade. He didn't think it had to be this way. Stallman believed that companies could still turn a profit without withholding the source code from the users. Furthermore he believed that it was the users right to control the software they used, and that no developer or company had the right to infringe upon that. In order to codify these rights, Stallman defined four freedoms, additionally he dubbed any software which followed these freedoms to be "Free Software."

The first freedom is "The freedom to run the program as you wish, for any purpose."[1] Motivated by the fact that the user should determine how the program is used, not the developer, no matter how strangely they wish to use it. If you wish to make YouTube only show you one video a day, you should be able to, regardless of what Google thinks.

The second freedom is "The freedom to study how the program works, and change it so it does your computing as you wish."[1] In order for this to be true, then the source code for the program must be freely available. This is motivated by the idea that the user should know exactly what a piece of software is doing on their system, and should be allowed to modify it if they see fit. Imagine if one of your programs makes all similar programs run slower, deceiving you into believing it is the fastest, then you are a tool to that software. The method to remedy this is for you to be able to fully understand that program.

The third freedom is "The freedom to redistribute copies so you can help others."[1] This one does not follow as cleanly from the idea of developers controlling their users, but instead seems to come from Stallman's aversion to NDAs, and his wish to create a world where more people can have access to high quality software. If you view software in the same way we view recipes, it does seem to follow that if you own the software you are running, you should be free to distribute it to others as you see fit.

The fourth and final freedom is "The freedom to distribute copies of your modified version to others."[1] He goes on to say "by doing this you can give the whole community a chance to benefit from your changes." Which seems to be motivated by some pragmatic argument for improving software for everyone.

With these freedoms in hand, Stallman sought to liberate the world of proprietary software, and began creating Free Software that followed these rules. He created the Free Software Foundation to oversee this goal, and got to work on ”The GNU Project” intended to create an operating system, and all the essential tools someone would need to make Free Software.

In order to ensure these four freedoms would be enforced, Stallman took advantage of copyright. A licenses can define certain liberties, and stipulations on someones copyright. Free Software employs specific licenses to accomplish its goals. At the creation of the GNU Project, Stallman helped to create the GNU General Purpose License (GPL). Any project using the GPL gives everyone the right to use the source code for private use, commercial use, and permission to distribute and modify the code. But it comes with some restrictions, in any further distribution you must disclose the source code, keep the same GPL license, and state any changes you have made to the source code. In this way the license protects and extends the four principles Stallman laid out. If I find your Free Software useful for my program, then incorporating it into my project forces my software to become Free Software as well.

With his philosophy ready, and a license to enforce it, Stallman began creating Free Software through the 80s, which gained a lot of popularity. Free Software such as GNU Emacs, Bourne Again SHell, and GNU C Compiler all became widely popular with developers. Stallman’s intuition was correct as well, while he was making less money than most businesses, lots of people were willing to pay for specific support using GNU Emacs. Stallman was more than willing to take a salary cut to support these ideals.

Not only did users have access to the four freedoms when they used these pieces of software, they were also high quality. Eric Raymond in *The Cathedral and the Bazaar*, describes the software as ”Cathedrals, carefully crafted by individual wizards or small bands of mages working in splendid isolation.”[4] But, while these beautiful cathedrals were causing a storm, they were barely denting the proprietary world. While many developers adopted Free Software, few consumers made the switch.

It would not be the Free Software movement directly that changed the future landscape of software, but a byproduct of its philosophy. It was 1990, a whole decade after Stallman began his crusade, the internet was growing, and information was being passed around faster than ever. Suddenly you didn’t need to write your software to a series of floppy disks, instead you could send it online. This would change how software development dramatically, and it started in one corner of the internet. A student from the University of Helsinki posted about a new piece of software he was working on, a piece of the operating system called the ”kernel.” The name of the project was ”Linux” a combination of the word ”Unix” and the student’s name, ”Linus Torvalds.”

What is important to understand about Linux is that it would be developed in a way very different to how things had been developed in the past. Rather than Linus and a few coworkers building Linux alone, like the cathedrals Raymond described. Through the power of the internet, anyone was made welcome to make a copy of Linux, change it however they pleased, and show it to Torvalds. If Torvalds liked the change he would add it, otherwise he would reject it, and you were free to continue making changes to your own version, or start over.

Torvalds had been inspired by Stallman’s Free Software, and licensed Linux with the GNU

GPL, but the important part wasn't that it was Free software. What made it special was the hyper efficient, rapid development that came with not just allowing, but encouraging everyone to contribute to the project, with Torvalds curating it along the way [4]. This method of developing software, where the source code was open to everyone, would later be termed "Open Source."

It is important to note that Open Source is not the same as Free Software. While every Open Source project fives the user the freedoms that Stallman dreamed of, there was nothing stopping an open source project from using a different license, and restricting users liberties. In this way Open Source could completely miss the philosophy of Free Software.

This ability to avoid the four freedoms may have improved Open Sources adoption. As companies quickly realized they could not compete with Open Source projects, they began to integrate them into themselves. With no fear of their products being infected like the GNU GPL forces. Alphabet Inc. known for Google LLC was one of the first to adopt Open Source projects in 2004. "Open source has become a pervasive component in modern software development, and Google is no exception. We use thousands of open source projects across our internal infrastructure and products .. We firmly believe in open source and its ability to bring together users, contributors, and companies alike to deliver better software."[5] Google would go on to create Android as a derivative from Linux, which is a piece of open source software used in most modern smartphones.

The shift was clear. Companies, and developers alike were switching to the Open Source model. Whether or not it was because it allowed for the four freedoms of Free Software, or because it was a more powerful method of developing software than ever before we will likely never know, but as we moved into the 2010s, we had entered a new era.

3 The Era of Open Source and Services as a Software Substitute

By 2015 large tech companies such as, Google, Facebook, Microsoft, and Apple had all decided to embrace Open Source Software. This was certainly a win for User's liberties. With the source code available users were free to ensure the security, or modify the software to suit their needs. But the shift left many suspicious. Just a decade prior, companies like Microsoft had openly denounced Free Software[3], and now they were creating entire departments in their companies for it.

There were a few things going on here. First, was that Open Source projects were a large pool for these tech giants to grab from. GitHub, a website designed for Open Source collaboration, reported that in 2015 44% of licensed projects used the Massachusetts Institute of Technology (MIT) License[6] which allows anyone to take the code and do whatever they please with the software, without keeping the same license. This meant that if there was an MIT project which solved a problem Microsoft had in Windows, they could take the source code, tweak it a little bit and put it right into Windows. That was a great reason to endorse Open Source.

Additionally, Linux, which was developed in an Open Source manner was becoming the dominant force in industry for servers, and other general uses. All these tech companies wanted a hand in its creation, and it paid for them to be buddy with the entire open source community.

Finally, something more insidious was rising, something worse than proprietary software for user's liberties. All this time Free and Open Source Software had been concerned with programs that the user was running on their machine. But we began to have a new concept come up with the internet: Web Applications. These are pieces of "software" that you access over the internet through your web browser. And it meant that companies could embrace the morally righteous Open Source, without giving up the power they had come to expect from proprietary software.

Google Documents is a perfect example of a Web Application. You go into your browser, log into your account and you are met with an interface which easily allows you to write a document online and access it from anywhere. But the question we should be asking is who is the user? While you may be the person using the software, the software belongs to Google. This leaves you in the same place as proprietary software, controlled by the software you wish to use as a tool.

Going back to the freedoms that Stallman was looking at, you have none of them. Even if you had access to the source code, there is no way to verify that Google's servers are running that piece of source code. Similarly if you modified that source code, you could not upload that software to Google's servers and begin using it yourself. In this way you are not actually using the software, but instead using a service which is a sort of software substitute.

These Service as a Software Substitute (SaaSS) products brought with them even more powerful ways to control the user, such as collecting large amounts of data on them, which then could be used to socially engineer the user into doing specific things.

Most companies proceeded one of two ways. The first option was what Google and Facebook did. They cemented their existing SaaSS, and began expanding and creating new ones. This allowed Google and Facebook to collect data on their users, while offering their products for free. They then used that data to make money through advertising with other companies. This encourages products like YouTube which are designed to keep you on the platform for as long as possible, controlling you.

The second option was employed by companies like Apple and Microsoft. Who used Open Source to bolster their existing products, and began shifting their focus from consumer goods, to products for other companies. Most other companies would rather have the direct support of Apple or Microsoft than some unorganized group online. Both Apple and Microsoft attempted to break into the SaaSS world, though had little success, and decided to instead use their power as the companies behind platforms such as Windows and MacOS to continue making profit.

Now we're left with growing support for the Open Source community, while companies encourage use of SaaSS. We've definitely made a lot of progress since Stallman's printer incident 50 years ago, but if you had a similar problem with your printer you would be stuck these days. And to compound these issues a whole new frontier of "software" in the form of SaaSS has popped up, and none of our tools for dealing with proprietary software will work here. We need to think about what is next, and what we should do.

4 The next era

I do not expect the landscape of software to change dramatically for at least another 10 years, but the seeds for the next era are always planted long before they sprout. So it is important to speculate. The choices that you and I make about what software we use and support will determine what software we use next decade.

While I think the Era of Open Source software has dramatically increased how much people use Free Software, I think it has yet to be advertised correctly to the average user. The brand name recognition of Microsoft Word or Outlook inspires support, while Open Source competitors such as LibreOffice Writer, or Mozilla Thunderbird provide all features in a different looking interface.

This is especially true for users who feel like they cannot perform certain tasks due to high buy in. While Photoshop or Mesa are incredible pieces of proprietary software, that people who make art or animation for a career should probably stick to, hobbyists should consider trying software such as GIMP or Blender, which are some standouts in the Free Software world for image manipulation and animation respectively.

Secondly, we need to start fighting back against SaaSS. All the things people complain about in their modern life, such as spending too much time on social media, or wasting too much money on DoorDash are byproducts of the software controlling them. Replacing SaaSS solutions with Free, or Open Source software will lead to a relationship of user and tool, rather than user and master.

That said SaaSS have some gripping features which we cannot reproduce with Free or Open Source Software yet. For one, you usually have an easier time collaborating, like in Google Docs. It is easier to pick up where you left off, in something like Googles Slides instead of Power point. The pick up and put down nature makes the SaaSS solution convenient for people with multiple computers, or if you need a coworker to see the same thing you do.

Additionally, SaaSS makes use of data that only a company has. There is no Open Source solution to DoorDash because you require the infrastructure of drivers and restaurants. DoorDash could release an interface for programmers to make Open Source Apps from, thus removing the predatory nature, but there are no incentives for DoorDash to do so.

We do not have a great solution for these problems at the consumer level yet. SaaSS makes use of having another computer that you connect to, which simplifies a lot of these problems. But you can't expect the average consumer to host their own server. Instead we need to investigate more peer-to-peer options. Those are options where instead of having a server which everyone connects to to collaborate, you all connect to each other directly and collaborate. This solves some but not all of the problems, and I believe it is the single most important next step in helping users gain access to Free Software.

The second problem of infrastructures which companies have and users don't isn't something we can solve on the consumer level. Unless a business incentive or legislative policy shows up, then companies are going to keep their infrastructures to themselves. The two options here are if consumers switch to options with public interfaces for programmers to make Free Software for, or if someone can come up with legislation which either encourages or enforces the democratization of these infrastructures. Though I have a hard time endorsing the legislative route.

Ultimately it comes down to consumer choice. Many people will purchase ethically sourced meat at a premium, and the same can be done for Free Software. If users are willing to pay extra for Free Software over proprietary or SaaSS then companies will adapt. For me, it is worth it to use Free Software over proprietary, not only because I have the ability to modify it, and change it to suit my tastes, but also ensure my software continues to be a tool and not a master.

References

[1] R. Stallman, “What is Free Software?.”

[2] T. Williamson, “History of computers: A brief timeline,” 12 2021.

[3] S. Williams, *Free as in Freedom: Richard Stallman’s Crusade for Free Software*. O’Reilly Media, 1 ed., 2002.

[4] E. Raymond, *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O’Reilly Media, 1 ed., 1999.

[5] S. Vargas, “Open source by the numbers at Google,” 08 2020.

[6] B. Balter, “Open source license usage on GitHub.com,” 12 2021.