



Published on Free Software Magazine (<http://www.freesoftwaremagazine.com>)

Free software vs. software-as-a-service: Is the GPL too weak for the Web?

Preserving software freedom in the era of Web applications

By Gavin Baker

You've read the GPL's preamble, you can name the Four Freedoms, and you do your best to keep proprietary bits off our computers. But what's the future of free software in the era of Flickr, Google Apps, and Facebook?

What it means to be free

The term *free software* was defined by Richard Stallman. We know Stallman as the founder of the GNU Project, the author of the General Public License, and founder of the Free Software Foundation. Stallman defined free software this way: not by the price of the software, but by the freedoms it accords to its users.

Specifically, users of free software are free to:

- Use the software for any purpose
- Study how the program works
- Change the software, modifying the source code to meet the user's needs
- Redistribute the software, with or without modifications—to share the software.

Free software is also known as *open source*. Access to the source code of a program is a precondition of the aforementioned freedoms—specifically, the freedoms to study the program and to modify it.

So what are the characteristics of free software? Here are some of the more familiar claims:

- Free software is technically better: it's better code.
- Free software lowers production costs by reducing unnecessary duplication of effort.
- Free software is more secure, because with many eyes, all bugs are shallow.
- Free software is more customizable, because every user has the tools to adapt the program to their needs.
- Because free software is produced in the open, where everyone is free to share and build upon the best work, competition is fierce and switching costs are low.

By now, these facets of free software are familiar enough to sound cliché? Let's dig a bit deeper: what else does it mean to be free?

You're in control

Digital restrictions are incompatible with free software. Effective DRM depends on preventing the user from circumventing restrictions; you can't do that when everyone can see the code.

In addition, the ways that free programs store and exchange data are open. As with DRM, the developer can't hide anything from the user, because she has the source code. So, by its very nature, free software uses open formats and open protocols. (Of course, published specs make interoperability easier, but that's another story. In the worst case scenario, a developer wanting to interoperate has to dig through confusing code; that's still better than having reverse-engineering as the only recourse.)

This is the social importance of free software

The picture I'm painting is a world where users have control of the software they use and the data they create. This is what's attractive about free software—the principles beyond mere pragmatism.

Consider the crucial role of software and data in the modern world:

- To economic vitality and innovation;
- To formal education as well as lifelong learning;
- To information about our communities—about our government, environment, and schools;
- To culture and creativity;
- And to our personal and social lives—as an extension of our selves, and a tool to connect with those around us.

Seen this way, what user *wouldn't* want to have ultimate control of the device they use for all these deeply important purposes; and what user *wouldn't* want control of the data they create with it? This, beyond any questions of technical merit or fiscal penny-pinching, is the social importance of free software.

Your data, their machine

Most users of GNU/Linux and free software already understand this—at least in a vague, gut sense. But our understanding (if not the principle) is based on an unstated assumption. The assumption is this: *We are* ultimately talking about our devices, right?

What does it mean to have control of software if it's not running on my machine? What does it mean to have control of my data if it's stored on someone else's computer? These are the questions raised by the rise of Web applications.

To preserve the freedoms we value, we have to reconceptualize user control



Figure 1: places like this are where much of our personal data are now stored

Services delivered over the web which perform a function which interacts with the user—in some cases, even duplicating the functionality of programs we’re accustomed to running on the desktop—are increasingly common. It’s been called *software as a service*, or more cheekily, Google OS. If we as a community are going to preserve the freedoms we value in the era of software as a service, we have to reconceptualize user control of software and data, and look at the new context we’re in.

In a sense, the “new” context of Web applications is not new at all. Text terminals, thin clients: there’s plenty of history of users running programs that live somewhere else. But for the most part, we still think of the software as being close to the user (for example on the same LAN). To run software whose administrator you don’t know and with whom you have no contact, on machines that are not part of your organization and may well be located on the other side of the globe—there *is* something qualitatively new about this.

Web applications actually make it easier for users to switch from proprietary OSs

Before confronting the challenges of this environment, I’ll consider the opportunities as well. The prominence of Web applications actually makes it easier for users to switch from proprietary operating systems, such as Mac OS and Windows. You don’t have to worry about how to use Outlook, or how to migrate, if your email client is Gmail. You don’t have to learn a new address book if Facebook is your Rolodex. You don’t have to switch to a new photo management suite if your vacation snapshots are all on Flickr. In fact, retail giant Walmart recently began selling GNU/Linux PCs by Everex, whose gOS distribution adds icons for common Web applications like Gmail and Facebook right on the desktop. Koolu, the low-cost thin clients promoted by Jon “maddog” Hall, bundles Google Apps for its customers. These cross-platform applications provide familiar faces for new users making the switch.

Challenges of Web applications to free software

There’s no reason that Web apps can’t be free software—and many prominent Web apps, such as WordPress and MediaWiki, are. But there’s a loophole which slows the growth of free software among Web apps: a loophole in the GPL, the most important free software license.

The GPL has a well-known reciprocity clause, or copyleft: if a developer modifies a GPL-licensed program, pastes a bit of code from it into his own program, or links with it, that developer is now obliged to “share alike” by licensing his contributions under the GPL. This is the *quid pro quo* under which users are granted the freedoms of the GPL. But the condition of reciprocity only becomes active upon the act of distribution—and there’s the loophole.

Free software vs. software-as-a-service: Is the GPL too weak for the Web?

If I distribute a GPL-licensed program, with or without my modifications, I have to extend the GPL's freedoms to users who receive the program from me. But if I run a GPL-licensed Web app on my server, offering public access to the application over the Internet—but never redistributing it—I am under no obligation to share the code with its users.

If you think that the GPL's "share alike" clause has helped spur the growth of free software (and I do) then this should be troubling to you. In a world where software is never distributed to users, but only run on server farms, authors of free software have no mechanism to ensure that downstream users will be free to use, study, modify, and adapt their software.

AGPL closes the distribution loophole, but adoption seems slow

At least, they *had* no option.

A company named [Affero](#) thought of this problem a few years back, and received the FSF's permission to modify the GPL, adding a clause to require reciprocity for users accessing the program over a network. The FSF considered adopting this clause into [GPLv3](#), but [dropped it during the revision process](#). However, they did adopt the [Affero GPL](#) as a new FSF license, updating it with the improvements in GPLv3 and making it possible to link AGPL-licensed programs with those licensed under the GPL. Unfortunately, adoption seems to be slow: a few Web apps have begun using the AGPL, such as [Wikidot](#) and the FSF's own [stet](#), but the most prominent free Web applications haven't converted to the AGPL, at least not yet.

Software freedom in the age of software-as-a-service

There are other considerations for user control in the era of software-as-a-service. Even GPL-licensed Web applications have no obligation to divulge their mechanisms for storing data or communicating with other applications, since the source doesn't have to be public.

However, let's presume that you do have the source code for the program, due either to the admin's generosity or to the AGPL. You can install a copy of the program and run it on our yown machine. You can learn how data is saved, and how the application talks to other programs. There's just one problem: you don't have your own data.

When you store information on a remote server, there's just one problem: you don't have your own data

In this example, you have been using the hosted version of this software, be it Gmail, Facebook, or [Launchpad](#). Now, you can run it and control it. What you lack is oyour data. That's right: your data, which you created—and from which the host has likely been profiting, by targeting advertising to you—is not available to you. You can ask the FBI what information they have about you, but you can't ask Facebook what information they have about you. More precisely, you can ask, but they won't give it to you. (Yes, I have asked. "Unfortunately, the feature you are requesting is not currently available," Chad the customer support representative informed me.)

Not everybody operates this way. WordPress, for example, makes it fairly simple to export your data. But too often, this is not the case.

Free software vs. software-as-a-service: Is the GPL too weak for the Web?

Let me clarify: why does this matter? Services like Facebook have a tremendous network effect. Similar to software with proprietary file formats, users are locked in because all their friends use the same platform. So you we don't have data portability, the opportunity for meaningful competition is slim—which means one company controls our destiny. *Meet the new boss, same as the old boss.*

Conscientious consumers

So if you're being conscientious consumers when deciding where to spend your ad views and personal information, what should you demand?

1. **Free license:** You should ask that our Web apps be freely licensed, as you would for programs on our own computers.
2. **Source available:** Because you can't rely on traditional copyleft clauses, you should demand that the source actually be available to users. Preferably, the authors should use the AGPL, to preserve the freedoms of downstream users.
3. **Data portability:** Before you entrust anybody with our precious data, you should make sure that you can take it with you.

Credits

Image: "Server Room : 0" by John Tregoning. Copyright © 2007; used under the Creative Commons Attribution 2.0 license.

Thanks to GatorLUG and Florida Linux Show, where presenting on this topic helped me to refine my ideas.

Biography

Gavin Baker (/user/45127" title="View user profile.):

Copyright information

This article is made available under the "Attribution" Creative Commons License 3.0 available from <http://creativecommons.org/licenses/by/3.0/>.

Source URL:

http://www.freesoftwaremagazine.com/articles/free_software_vs_software_service
