Software Libre and Commercial Viability

Alessandro Rubini

rubini@linux.it

1. Copyright and Licensing

2. Software Libre and Commercial Viability

Fortunately enough, Linus' own project of world domination is going to come true pretty soon. The trend can be verified by checking how the press is behaving towards GNU/Linux solutions, by looking at how several educational entities are going to introduce free software in the schools, and by verifying our usual technical excellence.

Nowadays, in 1998 (yes, it's still 1998 as I write this), the main task that needs to be addressed is propagating the social and commercial implications of free software. While I greatly appreciated Russ Nelson's article on the July issue of LJ, I feel the urge to expand on the points he touches.

Please note that I'm not an expert in economy nor politics. I'm just a build-it-yourself kind of technical guy who is extrapolating from his own experience in the battle for survival, hoping someone else could adapt his/her genes to the new environmental conditions. Some of these ideas, on the other hand, have already been discussed with friends and/or the Free Software Business mailing list (fsb-subscribe@crynwr.com) which I joined after Russel's article advertised it.

2.1. Viability for individual consultants

The best feature of computer systems is their flexibility, the ability to tailor the computer's behaviour to the user's needs. This flexibility is often completely unknown to the general computer user, because proprietary software solutions tend to hinder any functionality behind a rigid external interface which denies any divergence from the expected behaviour – user's behaviour.

When adopting free software, users are able to face the real power of computer systems. Today I talked with a commercial consultant who never thought that programs could be adapted to one's needs; he confessed that his company has always being acting the other way round: they adapted the needs to the software they use. Most users are victims of their software, and don't even realize it.

Educating the user base about the extendibility of software will open new markets to the independent consultants, creating new employment opportunities. Every user has different needs, and solving these needs often means calling technical support: people who tailor or enhance the relevant software. While this is not even imaginable with proprietary programs, source availability allows to quickly solve any problem that might arise and easily add new features. While you may object that this will quickly lead to a "perfect" software package, individual needs are so different and specialized that the catch-all package will simply not catch.

For example, I and others wrote a program for a local Physiology center, to analyze data for a typical kind of experiment. During two years of use, the physicians found so many ways to enhance the program that it is now reported as better than the commercial solutions. If you add all of the fees they paid during these years, the program reveals more expensive than some of the commercial alternatives; this fact is however irrelevant to my clients as they have exactly what they want, and they know they can have more should the need arise. The program is obviously GPL, and other centers expressed interest in getting a copy.

As more and more people are choosing free software to address their needs, I'm pretty sure some software companies will try to demonize Linux and the open-source movement because they are losing market-share. Such companies will probably try to demonstrate that IT employment is decreasing and the human kind is being damaged by the general adoption of free software. The whole argument, however, is bogus; computers exist to be programmed, and the more you allow programming them the more you build employment positions. If you count the number of people that offer free software consulting, you'll greatly exceed any shrinkage reported by proprietary companies. Sticking to the previous example, the Physiology lab hired my company to write the program, and other centers interested in the product are willing to hire a local consultant for installing maintaining and enhancing our package. Did I say "enhance"? Isn't the program working? Yes, the program is working well but there is room for making it better. The local lab decided to stop development "because we must run our experiment rather than invent new software features". As anyone knows, every program has a bug and a missing feature, and this is where we build our credibility: bugs can be fixed and features can be implemented. As I suggested earlier, the more you make things programmable, the more they will be programmed.

But why should there be more employment opportunities in IT than there are now? First of all because free software users have more requests for new features than users of proprietary products, as explained above. Next, because anyone can build her own professional skills without paying any tributes in order to access relevant information. That's how I built my Linux expertise: by studying source code and trying things out on my own low-end PC. Now I am confident I can solve anything my clients ask, and my clients know I can solve anything (provided I am given enough time to deal with the problem).

Another critical point, in addition to source availability, is standardization on file formats, a field where proprietary products are revealing their worst features. Let’s imagine an environment where every file format in the system was known: you could, for example, create indexes from every file that gets produced (document or wahtever), thus easing later retrieval. This can be accomplished off-line, without any load on the non-technical personnel. Asynchronous reuse of data is rocket science for many people, because they are assuefated to programs that use proprietary file formats (and operating systems with no real multitasking nor "cron" capabilities). As soon as free standards get adopted, users ask for customizations, and are willing to pay for anything that can increase their productivity. Moreover, free standards guarantee that customers are not making the wrong bet, as they won’t ever be stuck with unusable data if the software market changes.

Whereas the conventional model of software distribution accentrates all the knowledge in a few companies (or one of them), open standards leverage technical knowledge to anyone willing to learn. While a proprietary product can only be supported by a limited number of qualified consultants (whose number and quality is centrally managed), the number of consultants supporting a free software solution is virtually unlimited, and the offer can quickly adapt to the request.

In a world where computers are just tools to accomplish some other goals, easy customization and quick maintainance are basic requirements of power users. In my opinion, free software will quickly gain the trust it needs to be a real market player. As soon as you give some trust to open-source products, they reveal to deserve more, and GNU/Linux fans must be ready to offer support in order to fulfill the upcoming need for consultants.

2.2. Viability for support companies

Obviously, independent consultants don’t cover all of the needs of computer users. There are several activities that can’t be handled by individuals. Red Hat and Suse are demonstrating that creating and maintaining a distribution can be a good source of revenues, even if the product is freely redistributable; Debian-based efforts are in the way as well, although less advanced.

In addition to the “creating” (or collecting) and “packaging” jobs, open source companies can specialize in technical support, covering customers where computer systems are of critical importance. Big businesses that run computer systems in their productive environment won’t be safe enough with either the external consultant or the in-house technician. They need to be backed by an external structure that can guarantee 24/7 operation of their technological tools.

Even if GNU/Linux or any other operating system is demonstrated as completely reliable, power users will need to rely on a support company as a form of insurance. The more computers are important for a production environment, the more people is willing to pay for being reassured that everything will go on working, and for being able to shout to someone “responsible” should anything fail. Such a “power user” support contract could also include provisions for refunding the customre in case of down time; big support companies will be able to efficiently deal with it, and the clients will be happy to pay high rates if they don’t ever need to call for assistance.

In short, I see no need for software companies to have exclusive rights to any product, as the support services are relevant enough to offer good business positions in Information Technology. And who wants to be at the top could use some of his/her revenues to pay for free software development, thus granting access to the best software before anyone else. Such a patron will also be able to associate his/her name to software products. As a matter of facts, this practice is already pursued by the big distributions.

2.3. Viability for education centers

Needless to say, schools and universities have the best interest in teaching information technologies using free software tools. Due to its technical superiority, free software environments have more to offer to the students, but may also need more technical knowledge to be proficiently administered. I see no money savings here in choosing free operating systems over proprietary ones, but educational entities should better spend in hiring system administrators than feeding some already-too-whealthy commercial company. While my country is stuck with some rules that reserve money for buying things rather than money for raising human resources, other countries are moving in the right directions. Mexico and France, for instance.

There's one point more that pushes for free software in education: when students get a job, they tend to prefer tools they learnt at school, so they minimize extra learning efforts. This fact should make educational agents teach impartial tools (i.e., those not owned by anyone, those that are libre). Schools should teach proprietary software only if two conditions apply: there is no viable alternative, and the company that distributes such software pays the school for teaching its product. Paying some company to massively advertise its own product is definitely a non-sense.

2.4. Social issues

There are a few social issues related to choosing one software model or another one. Although I mark them as social, they have economical implications as well.

While free software may be no cheaper than proprietary software if you bill for your own time, different environment feature different rates in comparing time to money. Most emerging countries have good intellectual resources but little money, and they usually have many not-so-new computers as well. Proprietary OS's are unaffordable for them, but free solution are viable and productive. As a matter of facts, the already famous "halloween" document supports my point by noting that in the Far East Linux is growing very fast. Charity organizations usually feature the same environment: little money and a good amount of human resources. This leads straight to the free software model for their IT systems.

These ideas will probably suggest that free availability of information looks pretty leftist in spirit, as this "information to the masses" idea looks pretty similar to the reknown "power to the masses". What gets usually underestimated is the strong rightist flavour of the open source movement. The free software arena is fiercely meritocratic and is a perfect environment for free competition, where the laws of the market are applied to their best – only the best ideas and the best players are going to survive. Proprietary standards, on the other hand, tend to decrease competition by discouraging innovation and consolidating previous results.

2.5. Limits of the free software model

Naturally, I'm aware that not every software package can easily be turned to free software. I'm not talking about office products: I'm pretty confident some good projects will be fulfilled, sooner or later.

I'm rather considering all the environments where there is competition for market-share of products that are only loosely based on their software component. For example, an industrial equipment might include a computer and some commodity hardware (a robot, custom I/O peripherals, PLC's, ...); the software application hosted in the computer system is a minor part of the whole, but its features greatly affect the overall value of the equipment. Producing and debugging such applications usually requires huge investments, therefore preventing free redistribution of source code, as a form of protection against greedy competitors.

Another good exmaple is brought in by cell telephones. They include a lot of software, and such software is the single component that shapes the capabilities of the overall device, even though software is almost invisible to the end user, who perceives the device as a telephone rather than a computer. Such software is strictly proprietary because of its major functional role in the device.

Unfortunately, I see no easy way to liberalize this code. Although I don't care too much about cell phones (I don't use them :), I would better see free industrial applications because their technological content is usually worth reusing and adapting to new problems.

3. About the Author

Alessandro lives in one of the least Linux-aware towns in the least Linux-aware country in the world. He writes free software for a living and advocates free software for a mission. He hopes his upcoming child will keep off computers, recalling the good old times when such beasts where confined to their technical zoos. He reads email as rubini@prosa.it, trying to kill spammers and to reply to everyone else.