The Competitive Advantages of Free Software*

Alexandre Oliva <aoliva@redhat.com>
http://www.ic.unicamp.br/~oliva/

v1.14 of 2002/07/19 08:49:07

Abstract

This article proposes an analogy between organic life forms and software packages. It uses Charles Darwin's arguments of natural selection, such as increased variability and ease of reproduction, to support the claim that Free Software enjoys important competitive advantages over proprietary software.

Prologue

I had just finished reading an interview with Jeremy Allison [1] by ChangeLog, the Japanese subsidiary of LWN, when I called up my parents to tell them some news about my professional life. As I spoke to them on the phone, the interview I'd just read kept ringing in my mind.

I've known Jeremy for some time, from working with him in the Samba team, and I've enjoyed the fact that he earns his living writing free software. I also enjoyed the fact that I would have the same opportunity; it just so happens that I was calling my parents to tell them I had just been offered a position at a Free Software company, myself. My parents were curious about what Free Software was all about. Since I had never had such a good opportunity to evangelize to them, I couldn't miss this one.

Trying to find a good analogy to explain to my parents —both doctors— why Free Software has increasingly gained more acceptance, I recalled the interview I had just read.

In the interview, Jeremy had said:

> "I think you'll see once open source software colonizes a niche, it will completely dominate it."
>
> Jeremy Allison [1]

Thinking of this, a complete argument relating Free Software with (my views and recollections of) Darwin's theory of natural selection [2] formed in my mind and I presented it to my parents. Since the argument seemed quite successful in convincing them about the advantages of Free Software, and how a company that gave away the software it wrote could succeed, I thought I'd write it up and share it with everybody else.

1 Introduction

> "[...] dominant forms which spread widely and yield the greatest number of varieties tend to people the world with allied, but modified, descendants; and these will generally succeed in displacing the groups which are their inferiors in the struggle for existence."
>
> Charles Darwin, [2, chapter 11]

Compare the preceding quote with Allison's assessment of Open Source in a niche. Could it be that Jeremy had just read Darwin when he spoke his words? I believe that the similarity between these two quotes is not just a coincidence. In fact, I propose that Open Source Software [3] and Free Software [4] enjoy major competitive advantages over proprietary software, even though they are not (yet) the dominant forms.

This paper draws correlations between concepts of software development and of natural sciences. For example, a program shall be considered a species, the release of a program correlates with the variant of a species, and a copy of a program maps to an individual within a species. In addition, users' requirements for a piece of software define the niche of a species, and the user's CPU cycles and other machine resources correspond to the natural resources that living beings compete for. Software developers and maintainers show up as Mother Nature, the hidden forces of nature that introduce variations in species.

2 The Variability Advantage

> "[...] variations, however slight and from whatever cause proceeding, if they be in any degree profitable to the individuals of a species, in their infinitely complex relations to other organic beings and to their physical conditions of life, will tend to the preservation of such individuals [...]"
>
> Charles Darwin, [2, chapter 3]

Unlike natural species, software variations are not random, but are created by developers to adapt the software to users' needs. In fact, in the case of Free Software, it is the user that drives the variation process, by patching the software herself, by hiring someone else to do it or just by posting the change request to a mailing-list and hoping someone will volunteer to do it, for free or for a fee. It should be obvious that the freedom to make changes and to redistribute the new variations increases the variability of the species, making it more suitable for other, even unforeseen, niches.

Although such modifications do not need to be contributed back, they often are. When they introduce a feature useful for a wide variety of niches, a feature that represents a general competitive advantage over the original release, the changes are often accepted by the package maintainers, and quickly proliferate. When they are not generally advantageous, they are most likely to be rejected, and perish due to lack of maintenance resources or survive only in the limited niche for which they were designed, in which they are maintained separately from the main project.

Even when a change is introduced by the maintainers, users all over the world will be validating the changes around the clock, if the package is developed following the Bazaar model [5], i.e., with anonymous CVS [6] access and open mailing lists for public discussion. If a user disagrees with a change, he may complain and try to convince the maintainers to withdraw the change he didn't see fit. If the maintainers don't take it back, the user may keep on running the unmodified copy. Even though developers drive the introduction of mutations, they often do so based on user feedback, and it is user adoption that grants a variant more running copies at the end, so users play a central role in the process of selection of variants with the most appropriate features. As in nature, mutations of software perceived by users as advantageous for the niche they determine are more likely to survive and to be promoted by the users so as to leave descendants.

Proprietary-software companies, in comparison, choose a niche for their software in advance, and they won't allow a user to adapt the software to other niches. Even when they can afford to have many developers and are able to sell a large number of copies, their software still falls victim of reduced variability. That's not just because users can't make changes: software maintainers, proprietary or not, tend to consider variation bad, because, for each variant, they incur additional costs in maintenance and user support.

Free Software is not exempt from these costs. Sometimes, conflicts among members of a community of developers and users lead to a fork. Under a biological frame of mind, forks might be perceived as good, because they increase the variability of a species, while still allowing the sharing of code. For example, security fixes quickly propagate to all of the various BSDs; a port of GNU Emacs to a new operating system can often serve as a basis for a port of XEmacs, and vice-versa.

However, it takes a significant effort to maintain forks. They waste not only developers' time, as each feature is re-implemented over and over, but also community resources for testing and running the program, reducing the strength and coverage of each variant. Unless branches of a fork focus on different niches, as in the previously mentioned cases, it may happen that one draws sufficient attention so as to dominate the niche, displacing the others. Sometimes, before such a displaced variant dies, it is unified with the mainstream, as happened with EGCS/GCC.

A point worth mentioning is that software is intentionally, not randomly, adapted to new niches. One might even claim it would be better modeled after Lamarck's theory of evolution [7]. Lamarck believed that a living being inherited from its ancestors characters acquired during their lives. But Lamarck's theory does not apply to the analogy I propose. When a program is modified, it is not the original copy that acquires the change and then propagates it to its descendants. On the contrary: a program modification creates a new individual with a mutation, and the mutation may then be inherited by descendants of this new individual, as in nature.

3 The Reproductive Advantage

> "A great amount of variability, under which term individual differences are always included, will evidently be favourable [for the production of new forms through Natural Selection]. A large number of individuals, by giving a better chance within any given period for the appearance of profitable variations, will compensate for a lesser amount of variability in each individual, and is, I believe, a highly important element of success."
>
> Charles Darwin, [2, chapter 4]

Even though variability is the main theme in natural selection, Charles Darwin concedes that a large number of individuals may compensate for some lack of individual variability. Free Software, unlike typical proprietary software, can be freely copied, not only in terms of freedom, but also in terms of cost. Even though the GNU GPL [8] does not require anyone to give away software for free, someone who legally obtains a copy of the software is entitled to do so. This means Free Software individuals have an enormous potential to leave many descendants, tending to proliferate.

Proprietary software organisms, in comparison, only breed in confinement, and most often solely by means of cloning. The only situation in which proprietary software is not cloned is when a new version of the software is released. Given that such pieces of software aren't released very often, because of the increased maintenance costs, a large number of copies would be necessary for a software release to enjoy a favorable position over software whose individuals present a higher amount of variability. The immense number of copies explains the difficulty in displacing MS-Office in its marketplace: because it is so much widespread, users fear compatibility problems and difficulties adopting alternatives, therefore they keep using MS-Office.

It might be argued that Free Software for the masses, with binary distributions of packages accompanied by source code that most people don't use, could fall prey to the same problem of low variability. It is certainly true that the variation is lowered, but if enough people make use of the freedom to see and modify the code, they may keep the variability levels of Free Software higher than those of proprietary software, and the combination of high variability with increased reproduction rates may be able to offset a huge number of cloned individuals.

Besides Free and proprietary software licenses, there are other varieties of licenses worth mentioning. Most Freeware allows unlimited copying and use, but no modification, because, in general, source-code is not provided. Note that Freeware and Free Software are very distinct concepts: Freeware most often comes only in binary, pre-compiled form, generally for MS-Windows. Freeware suffers from the same cloning problems as proprietary software.

Other licenses do include the sources, but do not allow them to be freely redistributed, modified or not, which is almost like selling sterile software. One example is the Sun Community Source License [9], that allows modification for research purposes, but requires a separate commercial agreement for distribution.

Microsoft Shared Source [10] doesn't enjoy the benefit of increased reproduction rates, since it disallows copying, nor of the increased variation, since it forbids modification. In these senses, it is no different from a proprietary closed-source software license, and it does not benefit the software they cover any more than a closed-source license does. In fact, it may even hamper the development of alternatives, since anyone who ever looks at source code licensed under such strict use terms is tainted for life, risking being sued for copyright violation should they ever create similar software.

The issue of procreation only in confinement may not seem a disadvantage at first, given that it allows for man's selection, and man's-directed selection can be much more effective than random natural selection over short periods of time. However, it must be pointed out that the man that chooses the features of the software to be released, in this case, is not the user, but the company that produces the proprietary software, that wishes to hold control over the software and its users. For example, a company may introduce changes in their software with the sole purpose of making it difficult for competitors, that don't have access to their code, to create software that competes or inter-operates with it.

As in nature, the excessive dominance of a life form may prevent the survival of new life forms that attempt to occupy the same niche, even if the new life forms are better-fitted for the niche. However, if the new life form survives long enough to establish itself in a niche, increased reproduction rates and advantageous variations favor the domination of the niche by the new life form.

4 Conclusion

> "It may metaphorically be said that natural selection is daily and hourly scrutinising, throughout the world, the slightest variations; rejecting those that are bad, preserving and adding up all that are good."
>
> Charles Darwin, [2, chapter 4]

Doesn't it seem like Charles Darwin is talking about software development with the sources available for Anonymous CVS, with people all over the world downloading and testing up-to-the-minute versions? In general, Free Software developed in the Bazaar model evolves precisely as described in the quote above. Proprietary software, however, cannot enjoy the benefits of the fast turn-around time and the world-wide scrutinizing of every single change to the source code base, so it cannot adapt as easily to the moving targets that users' requirements tend to be. Of course, in either model, changes can be introduced that are disliked by users, but in the Bazaar development model, the influence a user can have in the selection of variants is much higher: not only can she choose a variant that doesn't contain the change, but also she can introduce further changes that adequate the software to her needs. Therefore, the Free Software model is more likely to fulfill users' needs, even though it may demand more effort and involvement from users to do so.

The arguments in this paper are based on my personal experience with Free Software development. They explain some of the reasons why "once open source software colonizes a

niche, it will completely dominate it" [1], especially when the Free Software is developed in the Bazaar model. It is not just a matter of good will, politics or philosophy; it's a matter of survival: "Survival of the fittest" [2, chapter 4].

Acknowledgements

I wish to thank Jeremy Allison and my parents for the inspiration; Islene Calciolari Garcia and Jorge Stolfi for the long discussions about this subject; Richard M. Stallman, for supporting the idea of writing this paper, for having reviewed it and mainly for having started the whole Free Software movement. I'd also like to thank Doug DeJulio and Jim Kingdon, and especially Paul Gallagher, Tom Tromey and Mary Lynn Kostash for the very careful reviews and the many useful suggestions to improve the paper. I claim full responsibility for any errors I may have insisted in introducing.

References

[1] Maya Tamiya, Jeremy Allison, and Takaaki Higuchi. Changelog interviews Jeremy Allison, December 1999. `http://lwn.net/2000/features/lc99/jeremy.phtml`.

[2] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.* 1859. `http://www.infidels.org/library/historical/charles_darwin/origin_of_species/`.

[3] The Open Source definition, v1.8. `http://www.opensource.org/osd.html`.

[4] What is Free Software? `http://www.fsf.org/philosophy/free-sw.html`.

[5] Eric S. Raymond. The Cathedral and the Bazaar, August 1999. `http://www.tuxedo.org/~esr/writings/cathedral-bazaar/`.

[6] Karl Fogel. *Open Source Development with CVS.* Coriolis, 1999. `http://cvsbook.red-bean.com/`.

[7] Jean Baptiste Pierre Antoine de Monet de Lamarck. *Philosophie Zoologique, ou Exposition des considérations relatives à l'histoire naturelle des animaux.* 1809. `http://gallica.bnf.fr/Fonds_Frantext/T0088740.htm`.

[8] GNU General Public License, v2, 1991. `http://www.fsf.org/licenses/gpl.html`.

[9] Richard P. Gabriel and William N. Joy. Sun Community Source License principles. `http://www.sun.com/981208/scsl/principles.html`.

[10] Microsoft Corporation. Microsoft shared source, May 2001. `http://business/licensing/sharedsource/default.asp`.