Overtaking Proprietary Software Without Writing Code

Proceedings for the 24C3

Overview

This is a brief summary of a 45-min talk aimed at software developers, with the aim of giving rough essential insights on how to overcome proprietary software. The key idea is that it is necessary to *look away* from pure code writing, in order to strengthen free software enough that it overtakes proprietary (non-free) software.

Part I: market overview

A brief reminder that although free software outperforms proprietary products in many respects, it still remains a minor player in the market. We develop the most stable, trustworthy, usable software in the world, and yet we fail to get past the 1% mark almost everywhere.

Perhaps most telling is the success of Microsoft Vista, whose supposedly poor performance we love to describe. In the first month of sales, Microsoft sold 20 million units. That's more Vista sales in one month than there has been GNU/Linux users in ten years.

So it's possible that we lack something to make a difference, and clearly it's not “good software”.

Part II: Obstacles

If we are to make a difference we have to solve or get around four problems.

1. Nobody chooses software

This fact is often forgotten because we typically are people who care so much about software that we build our own. But in our society our consumer lives are getting so impossibly complicated (there is a decision to make for just any purchase, from potatoes to batteries) that by the time they come home in the evening people don't want to worry about software. We have to be already “inside” when Joe buys his computer.

2. We'll never have a killer app

Because of the nature of free software, ideas and code flow quickly and we typically will never have a killer application (they get ported too quickly). We continually forget about this, however, and keep trying to build it anyway (ie. trying to make *the perfect, ultimate unique* application).

3. The legal environment is hostile

This is summed up in one sentence: in most countries you cannot play MP3s and DVDs with free software, legally. The code is here but the patent/DRM laws prevent using it legally. Until this is changed, free software will never make it to the shelves of any large-scale store.

4. The OS is disappearing

Because online services are typically well-designed, practical and sexy, we are losing hold of the “real” operating system. There will always be software needed to run the PC chips, of course, but all of the *interesting* software, with which we exchange ideas, produce work, and build our culture, is all progressively being transferred to private servers. Just ask how many people in a room full of developers regularly use Google apps, and how many use proprietary-software-devices to access some kind of closed network (in their car, pockets, or living room).

Unless we put our focus out of personal-computer-centric software, we are at risk of missing this change in computing trends.

Part III: Fundamentals

Making a real difference in the market means “tackling Joe”, the everyday user who has better things to do than worry about the status of his software's code repository. Two points here:

1. Talk to Joe. The fact is our community is so much focused on software stability and choice, that we shut ourselves on an entirely different planet. Perhaps insisting more on usability, absence of viruses, and simple, easy choices (ie. killing Distrowatch) is the first thing to do.
2. Be relevant. Source code is the least of concerns for 95% of users out there. Speaking of “free software” instead of “open-source” makes much more sense and does make a big difference whenever the Joe has to make a decision.

Getting back to basics, speaking a language that is relevant to Joe, is the sole focus of *GNU/Linux Matters*, a non-profit which aims to explaining Linux and free software to 1 million people in 2008.

Part IV: Breakthrough

The goal of this section is to introduce some “business-thinking” into software development. Because our software is available at no cost, we fail to think in terms of *market*, *customer expectation*, or *segmentation*.

On the proprietary side, knowing exactly what the consumers want and how much they are ready to pay for it is a priority. The products then stem *from* this analysis (for example, the various Vista or Photoshop versions).

In the free software world... we are often simply too busy forking to worry about what the users want. This is because of *The v0.12 Syndrome*, whose symptoms are **1.** A total dedication to quality (“the bug tracker *is* the project”) **2.** An agenda driven by the progression of the software (instead of the opposite, ie, “it's released when it's ready”) **3.** An overwhelming tendency to fork (whenever somebody disagrees on how the code is written). The result: high quality, stable software that's perpetually in a v0.12 state, and ten miles of altitude separating developers from users.

We'll start to break through when we realize that quality never has been a decision factor for the end-user. For example, OpenOffice.org is bloated but seduced 100m users (and is a major player in opening standards) because of good market analysis: being *just like MS Office* was the requirement there. Similarly, the only difference between Firefox and the low-profile Mozilla suite was some wise market analysis – a few cuts and some branding, not better quality, has made all the difference.

Concluding remarks:

Making a lasting dent into the overwhelming domination of proprietary software in the market does not require writing better code. What we lack is better market analysis: a more *tactical* perspective in the development of our projects, and a focus on what the users want. Giving up quality to work on differentiation, and adapting to the online world are two of the biggest requisites for that.

Talk given by Olivier Cleynen from *GNU/Linux Matters*, CC-BY-SA 2007. To learn more about us, visit http://www.gnulinuxmatters.org/ .