Cathedrals, Bazaars, and the Town Councils

Alan Cox

alan@redhat.com

These are some of my thoughts on the Bazaar model that I figure are worth sharing. It's also a guide to how to completely screw up a free software project. I've picked up a classic example of what I think is best dubbed the "Town Council" effect (although town councillors may think otherwise).

1. Copyright and Licensing

2. Cathedrals, Bazaars, and Town Councils

These are some of my thoughts on the Bazaar model that I figure are worth sharing. Its also a guide to how to completely screw up a free software project.

I've picked a classic example of what I think is best dubbed the "Town Council" effect (although town councillors may think otherwise).

There are certain things you have to understand about software developers. The first thing to understand is that really good programmers are relatively unusual. Not only that but the difference between a true "real programmer" and the masses is significantly greater than that between "great" and "average" in many other professions. Studies have quoted 30 to 1 differences in productivity between the best and the rest.

Secondly you need to understand that a lot of the wannabe real programmers are very good at having opinions. Many of them also catch buzzword disease or have some speciality they consider the "one true path". On the Internet talk is cheap.

The third part of any software project is what we shall call "the masses". They range between people who don't program but contribute massively in other areas - documentation, helping users and artwork to the sort of people that are often used to argue that you should require a license to connect to the Internet.

The project I'm going to take as an example of how to screw up completely is the Linux 8086 project. Porting a subset of Linux to the 8086 is one of the worlds more pointless exercises on the whole, and something that started as a joke and got out of hand.

There are a very small number of real programmers with the time and the right (or is that wrong) kind of mental state to contribute to a project whose sole real worth is "Hack Value". As a result of this at any given time the project has two or three core contributing people.

Unfortunately there are a lot of people who think it would be neat to run Linux on an 8086 who feel obliged to "take part". Most of them in this case are in the "wannabe programmer" category as the masses spotted the "silly" factor of the project from a safe distance.

The problem that started to arise was the arrival of a lot of (mostly well meaning) and dangerously half clued people with opinions - not code, opinions. They knew enough to know how it should be written but most of them couldn't write "hello world" in C. So they argue for weeks about it and they vote about what compiler to use and whether to write one - a year after the project started using a perfectly adequate compiler. They were busy debating how to generate large model binaries while ignoring the kernel swapper design.

Linux 8086 went on, the real developers have many of the other list members in their kill files so they can communicate via the list and there are simply too many half clued people milling around. It ceased to be a bazaar model and turns into a core team, which to a lot of people is a polite word for a clique. It is an inevitable defensive position in the circumstances.

In the Linux case the user/programmer base grew slowly and it grew from a background group of people who did contribute code and either had a basis in the original Minix hacking community or learned a few things the hard way reboot by reboot. As the project grew people who would have turned into "The committee for the administration of the structural planning of the Linux kernel" instead got dropped in an environment where they were expected to deliver and where failure wasn't seen as a problem. To quote Linus "show me the source".

If someone got stuck they posted questions and there was and is a sufficiently large base that someone normally has both the time and the knowledge to reply. In the Linux8086 case the developers had long since walled themselves off. Given a better ratio of active programmers to potentially useful wannabe programmers would have rapidly turned some of the noise into productivity. The project would have gained more useful programmers and they in turn would have taught others. As with any learning exercise you are better off having only a few trainees.

There is an assumption some people make that you can't turn the "lesser programmers" into real programmers. From personal experience in the Linux project there are plenty of people who given a little help and a bit of confidence boosting will become one with the best. There are many who won't but enough that will. [1]

The Linux 8086 project has mostly recovered from its 'infestation' and is now a small quiet project, using CVS trees and led by Alastair Riddoch who has been doing a sterling job. With the town councillors De-camped its now possible to ask questions, join in and help the project.

The lessons from this project, and others that went the same way (and sometimes died - remember the earlier Linux word processor projects) are fairly clear:

Release code right from the start. It doesn't matter if its not very useful. The best way to sort a town council is to simply do the job then tell them it has been done. Linux, KDE and GNOME have all taken this attitude and all done well from it. You can argue about the right way to program for a lifetime. Once there is code out there people (whatever their skill) can play with it.

Appreciate there are people who with a bit of help will contribute very much to a project. If their first patches are buggy don't put them down, explain why there is a problem and suggest solutions or places to look for examples of solutions. Every minute spent answering real questions helping someone work on a project will be paid back ten-fold to the project, and incalculably to society.

Don't forget non programmers. I find it sad that many people when asked "name the most important five Linux kernel people" rarely name some of the most important folk of all - the all to forgotten people who maintain web sites, change logs, mailing lists and documentation are as important.

Linus says "Show me the code". That is a narrow view of a real project. When you hear "I'd love to help but I can't program", you hear a documenter. When they say "But English is not my first language" you have a documenter and translator for another language.

Try and separate useful people from the noise. It is hard to separate people trying to help from a mass of pointless discussion and in the Linux 8086 case I definitely did the wrong thing by giving up on that goal. How to remove just those who talk and do not do anything is a research topic 8).

So next time someone wants to vote on a project, or discuss issues for a month and then implement it - be warned. They may end up with the right solution. The odds are however in your favour for carrying on regardless. Just ask them to send you a patch when it works.

Beware "We should", extend a hand to "How do I"...

Notes

1. As an example of this claim the original author of the Linux IPv6 code used to sit on irc from Portugal playing with a few basic ideas and asking questions. After we helped him figure some of the kernel internals he wrote probably 75% of the Linux IPv6 stack and was last seen working in the USA for cisco.