The Risks of Closed Source Computing

Alan Cox

alan@redhat.com

One large vendor likes to talk about the risks of Open Source software, but the strange thing is, the risk is actually in closed technologies. This article looks at the real risks in following a proprietary software path.

1. Copyright and Licensing

2. The Risks of Closed Source Computing

In the commercial world it is important for companies to reduce risk. One traditional approach to this is to use commodity parts and to demand that vendors work to those standards. In a mature industry there are standards,

standards bodies and interworking. If a company like Dell has problems with a hard disk supplier they shrug and use another one. They are protected from vendor lock-in and other related risks.

Over the past twenty years the computer hardware industry grew up. It became a commodity business. The IBM PC and the clones wiped out the minicomputer industry with ease. To business the case was simple. If you buy a minicomputer then the vendor gets to screw you for as much as they think you can pay for any upgrades. If you buy a PC you get to dictate terms to your suppliers.

The minicomputer vendors tried very hard to stop this commodity market from eating them alive. They called the PC a toy, they claimed it was only suited to low end tasks, that it would never be able to replace a real computer. They were wrong. They were stomped into oblivion by the power of competition, volume savings and commodity pricing. [1] You might of course recognise the rhetoric. They even published figures of dubious value showing that minicomputers were so much faster than a PC that the PC was dead.

Why is this relevant to Open Source? Well it is the same situation. No company now would commit to a closed hardware strategy. It would cost them more than using commodity components. Just as importantly, it would commit them to a single source for support and parts. Why then do they commit to a single software supplier ?

A closed source strategy exposes the company to serious business risk. As many telephony companies have discovered, your OS supplier might suddenely decide to be your competitor. It would be naive to think they pay the same price internally within the OS supplier that they levy for licensing in a competitors telephony products. No company can survive with a critical component totally controlled by a competitor. They will be systematically bled of money, and them stomped.

There is also no comeback in a closed source environment. Big name software vendors fill their licenses with clauses that both deny anyone else the right to deal with their software bugs, and deny the right of the purchaser to expect problems to be fixed. Absolute power is exercised over bug fixing. Are you now a competitor, did your CEO say something critical about the vendor ? Maybe that is why you can't get a show stopper bug fixed.

You may think this is a scare story but unfortunately it is not. Ask the companies who committed to Windows NT on the Alpha how they are feeling right now. Look at your own Y2K spending and see how much was caused by upgrading forced on you because you did not have the rights to fix just the Y2K bugs: a cost that would have been happily shared across thousands of companies, all of whom could have avoiding upgrading to bleeding edge products that demanded much more powerful hardware.

In an Open Source environment the software comes with rights to modify. The source code is not actually the important bit. It is a tangible artifact of the openness. The really important rights are the rights to modify and to see the workings of the system. If your vendor doesn't support you, then the rules of traditional business apply. You go to their competitors. If the software doesn't interwork with another package then all the information to resolve it is available. You are not locked in by a single supplier who refuses to tell you how to interface to a competitors product.

In a serious production environment every component is dual-sourced. Nobody running a production line can afford to risk their only supplier phoning up one afternoon and saying "Hi, we've doubled the price", or worse yet "XYZ bought us out, so we won't be supplying you in future". They have several suppliers or potential suppliers. If one demands too much money, they change. It may appear possible to have a single supplier and a tightly worded contract. Companies go bankrupt however. If a contract argument goes to court you may well be out of business before you supply of parts is resumed.

Similarly, in a production environment, downtime is unacceptable. Companies need guaranteed, strong, support. But they need something else: they need multiple sources of support. If the vendors recommend a maintenance engineer who is incompetent you have to be able to go elsewhere. If you buy a car and the local garage is useless, you take it elsewhere for repairs.

In many ways the motor car is a very good example of the fact that the open source model is not something revolutionary, as Bob Young is so keen to point out - it is the model we use in almost all serious grown up industry.

You can buy a car from several vendors. They are all basically similar. You get particular features from some vendors, and these vendors use perceived quality to persuade you to buy their car over another. In other words they have brand names and they have to offer something.

You can drive any vendor's car on any road. You can switch to another vendor's car without much relearning, as you can with a change of Linux vendor. There are no huge artificial and intentional barriers.

You can also fix your own car. As an individual you may want to do this and find it cheaper. You might also fix your own car and tinker with it because it is fun to do so. In a company environment you use words like "total cost of ownership" and you pay people to fix your cars. You can however choose just who you have fixing your car; or to buy a different sort of car in future without problems. If you have problems you move on to another supplier of the service - just like Open Source.

Another interesting parallel is that of accreditation. Most people do not know a good car mechanic from a bad one. Most non-computing people don't know a good software engineer from a bad one. Car manufacturers accredit companies and individuals they feel offer a high enough standard. They put their own name on the line to guarantee the quality of those they certify. If an official Ford dealer lets you down, you blame Ford, not just the dealer. In the Linux world you have things like the Red Hat Certified Engineer and you have approved support partners.

If you happen to know a good cheap mechanic then you can use them. People who find a good local car repair will often remain very loyal to it. If you happen to know a good local Linux support company and you prefer the local touch and being able to talk face to face with someone who actually remembers your case history you may well prefer that option. In an open source world they can do far more than just talk - they have the same access to source code and rights to fix it as the large vendors.

Many people talk about the Open Source advantages of price and of reliability. They actually miss the biggest advantage of all by doing so. Open Source is about commodity product in a grown up market. It is about competition, choice and putting power in the hands of the consumer.

In an Open Source world nobody gets to hold you, the customer, to ransom.

3. Credits

Much of this article arises from thoughts after listening to Bob Young's views on commoditization of software and discussions with Jamal Hadi Salim on the risks to large corporations of committing to a single source software product from a potential competitor.

Both Richard Stallman (who I hope will forgive me for using the words 'open source') and Eric Raymond have put forward aspects of these ideas. Eric tends to focus on the risk aspect and puts forward a model where open source is a matter of mutual self-interest. Richard puts forward a model where the focus is much more about co-operation. Which of the two is right (and indeed whether they are the same thing) is an interesting question, but not one relevant to this article.

Notes

1. As an aside to the main discussion, one reason that the Beowulf clusters have been so dramatically successful is that it is a set of open source software that allows the replacement of proprietary closed mainframe hardware with standard PC components. The Beowulf thus delivers a double blow against proprietary competitors. It breaks the existing mainframe lock-in and it breaks software lock-ins.